GRAFICOS AVANZADOS CON AMSTRAD

Robert Ransom



GRAFICOS AVANZADOS CON AMSTRAD

Robert Ransom



Publicado en España por:
RA-MA Editorial
Carretera de Canillas 144
28043 Madrid
Tfnos (91) 200 97 46 - 47

Publicado en el Reino Unido por
SIGMA PRESS
98A Water Lane
Wilmslow
Cheshire
United Kingdom

Título original en inglés:
AMSTRAD GRAPHICS - The advanced user guide

Impresion: SIGNO IMPRESORES, S.A.
Albasanz, 27
28.017 MADRID.

I.S.B.N. 84-86381-20-7
D.L. M-41.272-1.986

Traducción y composición: Manuel Baselga López

P R Ó L O G O

Este libro trata de los gráficos en los ordenadores domésticos Amstrad, en concreto el CPC 464, el CPC 664 y el CPC 6128. Estas máquinas están especialmente indicadas para los trabajos gráficos, por diversas razones. En primer lugar, la versión de BASIC contenida en su ROM es muy rápida, lo cual permite al usuario efectuar cálculos numéricos o dibujar líneas con mayor velocidad que la mayoría de los otros ordenadores domésticos. Además, la pantalla gráfica está dispuesta en un área de 640 x 400 unidades, resolución similar a la de muchos dispositivos gráficos profesionales (aunque, como veremos en el Capítulo 1, la resolución "verdadera" es en realidad de 640 x 200 pixels). Pero seguramente la principal ventaja de los ordenadores Amstrad en aplicaciones gráficas es la gran variedad de comandos gráficos que ofrece su lenguaje BASIC. A diferencia de los desgraciados propietarios de un Commodore 64, el usuario de ún Amstrad puede dibujar líneas, cambiar los colores de la pantalla y de los puntos, o incluso colorear zonas de la pantalla (sólo el CPC 664 y el CPC 6128) sin necesidad de manejar interminables ristras de PEEKs y POKEs en distintas posiciones de memoria.

Este libro pretende ser una introducción sencilla a los gráficos por ordenador para el programador no experimentado, además de servir como primer contacto con el mundo de los auténticos gráficos por ordenador: los que se utilizan en el diseño asistido por ordenador, trabajos de simulación y estudios artísticos. Gran parte de los aspectos básicos que se cubren en este libro se aplican en los sofisticados paquetes de *software* que se ejecutan en ordenadores de muchos millones de pesetas.

En las páginas siguientes el lector aprenderá a construir imágenes sencillas y gráficos mediante caracteres de texto, a dibujar y manipular objetos con estructura "de alambre" en dos y tres dimensiones, a dibujar en perspectiva, a emplear algoritmos de "líneas ocultas", a construir figuras a base de segmentos y a rellenar los diagramas de las estructuras para conseguir imágenes sólidas en alta resolución. Se ofrece una amplia variedad de programas-ejemplo; además, se configurará una biblioteca de subrutinas gráficas con la cual el lector podrá definir su propia "biblioteca gráfica", utilizando las mismas técnicas. Se supone que el lector ha escrito al menos algunos programas sencillos en BASIC, y está familiarizado con los comandos del BASIC de Amstrad. Los manuales de Amstrad son lo bastante detallados como para comprender las técnicas generales de programación que se utilizan en este libro; no obstante, si se desea ir más lejos en este aspecto, recomendamos el libro *Amstrad CPC 464 - Programación Avanzada*, de Mark Harrison, también publicado por Ra-Ma Editorial, y otros títulos de esta misma editorial dedicados a los ordenadores Amstrad.

Aunque la comprensión de los gráficos por ordenador requiere una cierta familiaridad con las técnicas matemáticas implicadas, es posible utilizar las rutinas sin disponer de conocimiento alguno de geometría del punto ni de álgebra de matrices. Ello es posible gracias a que el propósito de cada una de las rutinas está claramente explicado, junto con los detalles de las variables de entrada y de salida. Para el lector más avezado en matemáticas, se ofrece un apéndice que describe las manipulaciones de matrices que intervienen.

He intentado que el contenido de este libro sea accesible al mayor número posible de usuarios de ordenadores Amstrad. Los programas-ejemplo han sido desarrollados en un CPC 664 con monitor en color, y funcionarán exactamente igual en el CPC 6128. Entre el CPC 664 y el 464 sólo hay algunas sutiles diferencias, por lo que el usuario equipado con la máquina sin disco no tendrá dificultad alguna en este aspecto. Ni siquiera existe distinción entre el disco y la cinta - los comandos de entrada y salida del excelente intérprete de BASIC de Locomotive son idénticos para ambos.

Puesto que este libro se ocupa casi exclusivamente de los gráficos con el ordenador Amstrad, quizá desee alguna información sobre la forma en que han sido preparados los propios gráficos que ilustran el libro. Fundamentalmente, proceden de dos fuentes. Los listados de los programas fueron producidos por una impresora Epson MX 82F/T, conectada a través de un cable de interfaz Centronics al puerto de usuario del CPC 664. El *software* que pilota la interfaz se obtuvo de dos orígenes distintos: el primero es un listado en código máquina de F M Collins, que encontramos en el examen de Abril de 1985 de Informática Práctica (este programa necesitó algunas modificaciones para funcionar con la impresora Epson). Un programa más sofisticado de volcado de pantalla es el Tascopy, que se ha empleado extensivamente aquí. En el Capítulo 1 pueden encontrarse algunos detalles de este programa.

Algunos de los diagramas se han obtenido mediante un procedimiento bastante más serio. Para ello se ha empleado un "supermini" Digital Equipment Corporation (DEC) Vax 11/780, y los programas para generar las Figuras fueron escritos en Fortran 77, utilizando el *software* gráfico Tektronix Plot-10, que consiste en una serie de rutinas que pueden invocarse para efectuar operaciones gráficas. Las Figuras definitivas fueron impresas por un trazador gráfico horizontal de dos plumillas Tektronix 4663. La elección de este equipo (cuyo precio supera los 50 Millones de pesetas) se debe únicamente a su disponibilidad, además de a la consideración práctica de que este autor no disponía de ningún trazador gráfico conectado a su Amstrad. En el Capítulo 1 puede encontrarse una explicación más detallada de los merecimientos y prestaciones de las impresoras y trazadores gráficos.

El uso que cada lector dará a las técnicas de este libro depende en gran medida de las circunstancias individuales. Aunque mucha gente ya tendrá una idea de para qué quieren los gráficos, este autor confía en que las técnicas aquí presentadas sirvan para estimular el interés en las

posibilidades de los gráficos en los ordenadores Amstrad. Este libro puede utilizarse a varios niveles. Si lo único que se desea es disponer de algunos programas que generen efectos gráficos interesantes, con la sola intención de impresionar a las amistades, este libro puede ser una buena ayuda. No obstante, espero que muchos lectores desearán ir más allá de esto e intentarán comprender los principios básicos en los que se fundamentan los gráficos por ordenador. Podrá Vd. darse cuenta de que muchos de los programas aquí incluidos pueden mejorarse ampliándolos, extendiéndolos, o simplemente agrupándolos en "paquetes" de programas. Hemos intentado separar los programas en secciones independientes, procurando que éstas sean lo más sencillas posible, por lo que hemos evitado ese tipo de "*software* integrado" en aras de la claridad.

Permítame desearle buena suerte, y esperar que disfrute tanto explorando las capacidades gráficas de su ordenador como este autor disfrutó escribiendo este libro. Si localiza algún fallo o error, le agradecería que me lo hiciese saber, y si sus manipulaciones en los programas dan buenos resultados, bien, quizá desee comunicarme sus triunfos también. Ahora, con su permiso, voy a intentar asignar otros valores a los parámetros de esas curvas fractales...

Durante la preparación del libro he recibido diversas ayudas. Ray Matela fue el primero que me introdujo en el mundo de los gráficos por ordenador, y he aprendido mucho de su amplia experiencia. Debo agradecer también a Greg Turk su autorización para utilizar una versión adaptada de su programa fractal para el Apple (gracias también a Peter Sorensen), y a Tektronix Inc (Beaverton, Oregon) por su autorización para copiar las Figuras 7.1 y 7.6 del Manual de Usuario de su Plot 10 3D. Graham Beech, de Sigma Press, me animó a considerar los ordenadores Amstrad para trabajos gráficos (y me prestó un CPC 664 para desarrollar los programas de este libro). Por último, debo agradecer a mi mujer y a mis hijos el haberme permitido aislarme durante largas horas.

Robert Ransom

Woburn Sands
Julio 1985

C O N T E N I D O

INDICE DE PROGRAMAS

La lista que se ofrece a continuación se incluye como referencia visual rápida de lo que su ordenador será capaz de hacer una vez escritos los listados que aparecen en el libro. Todos los programas han sido verificados en ordenadores Amstrad, y funcionan correctamente. Obsérvese que algunos de los listados deben ser mezclados (MERGE) con programas principales cargados previamente en el ordenador. Estas mejoras se indican con un asterisco.

Capítulo 1

BLOQUE
Programa de demostración del empleo de bloques gráficos en baja resolución para crear una imagen.

COLOR
Muestra los colores disponibles en los ordenadores Amstrad como una serie de tonalidades de gris en la salida por la impresora.

HEXAGONO
Sencillo programa que dibuja un hexágono. Demuestra simplemente el ajuste de la pantalla de alta resolución y los comandos paara dibujar líneas.

CIRCULO
Otro sencillo programa, que en este caso nos muestra cómo una simple fórmula puede servir para generar un contorno regular.

ESPIRAL
Programa que genera una espiral.

GRAFICAS
Programa de demostración que puede utilizarse para dibujar gráficas. Las escalas de las gráficas, las etiquetas y los datos se introducen durante la ejecución del programa.

PANTALLA
Programa que marca las dimensiones de la pantalla para comprobar su tamaño en la salida impresa (si se dispone de una impresora, claro).

Capítulo 2

AHORALOVES
Este programa demuestra la escritura y borrado de líneas para producir el efecto de movimiento.

INVERSION
Programa que demuestra el efecto de invertir los pixels en lugar de escribirlos y borrarlos.

CRUCE
Comprueba los pixels para evitar el cruce de líneas.

DEMOMASK
Demostración del empleo del comando MASK en el CPC664 y CPC6128.

TRAZOS
Programa para el CPC464 que genera líneas discontinuas.

ELIPSE
Programa de generación de elipses.

SENO
Programa que genera ondas sinusoidales.

PARA
Programa que genera curvas parabólicas.

VECTOR
Programa que muestra la escritura y borrado de vectores lineales para producir animación.

FRACTAL
Demostración de los impresionantes efectos gráficos que pueden obtenerse representando curvas fractales.

Capítulo 3

DIBUJOFACIL
Programa que demuestra el empleo de estructuras de datos para definir datos bidimensionales.

FICHERO2D
Este programa permite crear ficheros que contienen datos de líneas y puntos para dibujos en dos dimensiones. Esta versión almacena los ficheros en disco, pero en el texto se explica cómo guardarlos en cinta.

DIBUJO2D
Versión extendida de DIBUJOFACIL que toma como entrada datos procedentes de un fichero de disco o cinta (creado mediante FICHERO2D).

SKETCH
Programa que permite la creación interactiva de un conjunto de datos bidimensionales, empleando un *joystick* para controlar el movimiento del cursor.

Capítulo 4

GIRO
Muestra el movimiento de una flecha para ilustrar las rotaciones en dos dimensiones.

TRV1, TRV2, TRV3
Estos programas se desarrollan a partir de GIRO, y se ponen a punto a lo largo del capítulo para manejar transformaciones generales en 2 dimensiones.

V1 rotación + traslación (flecha)

V2 rotación + traslación + cambio de escala (nave espacial)
V3 es igual que V2, pero lee los datos de un fichero secuencial creado mediante SKETCH y utiliza un método de multiplicación de matrices.

ZOOM
Demuestra el uso de un veloz algoritmo que permite efectuar el *zoom* hacia delante y hacia atrás de una imagen.

CUADRANTE *
Versión de SKETCH que permite dibujar un objeto de tamaño cuatro veces mayor que la pantalla. Se trata de un programa muy útil para confeccionar ficheros de datos que contengan mapas, etc, y se utiliza en conjunción con el ZOOMCUAD que se indica a continuación.

ZOOMCUAD *
Versión de ZOOM que manipula toda el área de datos de 1280x400 de que dispone CUADRANTE.

Capítulo 5

TARTA
Programa que genera gráficas de tarta: puede utilizarse en cualquier modo de pantalla.

PARTICIóN
Versión de TARTA que permite "partir" los sectores individuales de la tarta, separándolos.

MINITARTA
Versión de TARTA que permite visualizar al mismo tiempo múltiples tartas.

SUPERG
Versión expandida del programa GRAFICAS del Capítulo 1. Puede utilizarse para representar los datos por puntos o mediante una línea continua.

TABLA
Programa de gráficas que escribe los meses en el eje X.

ENFASIS *
Versión de TABLA que permite comparar dos conjuntos de datos.

ACUMUL *
Versión de TABLA que muestra una representación acumulativa de los dos conjuntos de datos.

BARRAS *
Versión de TABLA que dibuja barras en lugar de puntos de datos.

PATRON
Programa para sombrear rectángulos con diversas tramas

TRAMA *
Unido a BARRAS, permite tramar los patrones a utilizar en las gráficas de barras.

BARCOMP *
Versión de BARRAS que representa dos conjuntos de datos de barras en los mismos ejes.

HISTO3D
Programa que genera histogramas tridimensionales.

Capítulo 6

DISEÑO
Este es un programa de diseño asistido por ordenador que puede utilizarse en diversas situaciones en las que deban encajarse varios objetos en un espacio: la disposición de una habitación, el diseño de circuitos, etc.

Capítulo 7

FICHERO3D
Este programa es una versión extendida de FICHERO2D (Capítulo 3), y permite crear ficheros de datos en tres dimensiones.

SKETCH3D
Versión mejorada de SKETCH (capítulo 3), que permite la "creación" interactiva de ficheros de datos tridimensionales "en pantalla".

PROY3D
Este programa dibuja una imagen utilizando datos en tres dimensiones. Puede utilizarse bien con un fichero en 3D en disco o cinta, bien con las sentencias DATA del propio programa.

TRASL3D
Programa cuya base es la sección PROY3D, con una rutina de rotación tridimensional. En el texto aparecen también rutinas para el cambio de escala y la rotación.

PERS3D
Esta es una versión extendida de PROY3D que dibuja en perspectiva figuras tridimensionales con estructura "de alambre".

TABLON *
Este programa es una mejora de PERS3D que permite proyectar un conjunto de datos de dos dimensiones como un "tablón" de 3 dimensiones.

Capítulo 8

PINTOR
Programa que demuestra el uso del algoritmo del pintor para eliminar de una imagen las superficies ocultas.

FICHERO3DO
Versión de FICHERO3D que incorpora más sentencias de entrada para manejar los datos adicionales de superficies necesarios para dibujar una imagen con líneas ocultas.

OCULTAS
Programa principal de eliminación de líneas ocultas, que permite suprimir éstas de un objeto convexo simple, con el origen situado en el interior de este objeto.

S3DO *
Versión de SKETCH3D que añade al conjunto de datos creados información adicional sobre las superficies.

OCULTAS2 *
Esta versión de OCULTAS puede utilizarse en conjunción con S3DO para crear una imagen con líneas ocultas de un conjunto de datos "expandido" a partir de dos dimensiones.

Capítulo 9 MOL3D

Programa para dibujar moléculas que representa los átomos como esferas en el espacio de tres dimensiones.

ENTRADAMOL

Este programa crea el fichero de datos de las moléculas que debe leer MOL3D, y que contiene las coordenadas X,Y,Z para cada átomo, junto con los radios atómicos.

Apéndice 2

MULTIPLICACIÓN DE MATRICES 1
MULTIPLICACIÓN DE MATRICES 2

Dos rutinas para multiplicar matrices de 1x3 por 3x3 y de 3x3 por 3x3, respectivamente.

Capítulo 1

INTRODUCCION

1.1 ¿Qué son los gráficos por ordenador?

Los gráficos por ordenador son la representación visual de la información numérica codificada en el interior del ordenador. Aunque la salida normal de texto en la pantalla del ordenador es, en sentido estricto, "gráfica", los gráficos por ordenador empiezan realmente cuando se visualizan gráficas, histogramas, dibujos y animaciones.

Antes de que apareciesen los ordenadores domésticos, los gráficos por ordenador estaban reservados a los grandes ordenadores industriales y educacionales, cuyos usuarios podían invertir millones de pesetas en *hardware* y *software*. Hoy, sin embargo, los papeles se han invertido, y los ordenadores caseros poseen una gran ventaja frente a los grandes ordenadores: en la mayoría de los micros, las capacidades gráficas vienen incluidas de fábrica, tanto en *hardware* como en *software*, mientras que los usuarios de grandes ordenadores deben adquirir por separado programas y terminales sobre los que ejecutar los paquetes gráficos, y éstos no son en absoluto baratos. Por supuesto, los grandes ordenadores ofrecen la ventaja de la velocidad a la que pueden hacer las cosas y de la cantidad de memoria de que puede disponer el programador. En varios lugares del texto volveremos sobre este tema, para comparar los gráficos en ordenadores centrales y en micros, pero de momento echaremos un vistazo a los aspectos básicos de los gráficos por ordenador, antes de ocuparnos de los ordenadores Amstrad en concreto como vehículos de nuestros estudios gráficos.

1.2 Elementos de los gráficos por ordenador

El elemento individual que más ha favorecido el florecimiento de las capacidades gráficas en los micros caseros ha sido el desarrollo de una tecnología de visualización por puntos. En los comienzos de los gráficos por ordenador, las primitivas gráficas (puntos, líneas, áreas rellenas) se generaban controlando directamente el haz del tubo de rayos catódicos. Este tipo de pantalla (llamada pantalla vectorial) sigue utilizándose en gráficos profesionales que requieren alta precisión. Aunque son unos dispositivos muy divertidos de utilizar, no parece que sean lo más apropiado para los micros caseros. Ello se debe, sobre todo, a que son caros, y a que no suelen permitir borrar líneas de forma selectiva: solo puede renovarse la pantalla borrándola en su totalidad de una sola vez.

Más recientemente, han cobrado importancia ls pantallas refrescadas, que superan las limitaciones de las pantallas vectoriales. Ante todo, la tecnología de refresco es tecnología de TV, por lo que puede adquirirse un nuevo dispositivo de visualización monocroma por poco más de 15.000 pesetas. Además, el propio término "refresco" nos está indicando que la imagen es renovada constantemente: una televisión estándar lo hace 25 veces por segundo. Ello significa que puede eliminarse una línea y volverla a colocar en otro lugar distinto, mucho más deprisa de lo que el ojo humano puede apreciar. En realidad esto es más sencillo en teoría que en la práctica, puesto que es necesario poner a disposición del ordenador la información visual necesaria, para que éste opere sobre ella. Esta información visual se guarda en una zona de la memoria del ordenador conocida como fichero de pantalla, o memoria de pantalla. Esta parcela de la memoria es, en esencia, un "mapa" de los puntos visualizables. Cada uño de esos puntos se conoce como *pixel* (abreviatura de las palabras inglesas *picture element*), y cada pixel requiere una posición de almacenamiento en el fichero de pantalla.

En la forma más sencilla de un fichero de pantalla, se utiliza un solo bit para representar cada pixel (por eso suele aludirse con frecuencia al fichero de pantalla de los micros como un "mapa de bits"). Si el bit tiene valor 1, el pixel está encendido, y si toma valor cero permanece apagado. Esta representación es ideal para las pantallas monocromas sin tonalidades de gris, ya que cada punto sólo puede ser oscuro o luminoso. Si han de dibujarse varios colores o tonos de gris, habrá que disponer de más de un bit para representar cada pixel. Si alguna vez se introduce en el mundo de los gráficos profesionales, se encontrará con el término "plano de bits". Cada plano de bit es un bit extra utilizado para cada pixel. Dos planos de bits dan lugar a cuatro posibles combinaciones de colores, tres planos de bits a ocho, y así sucesivamente.

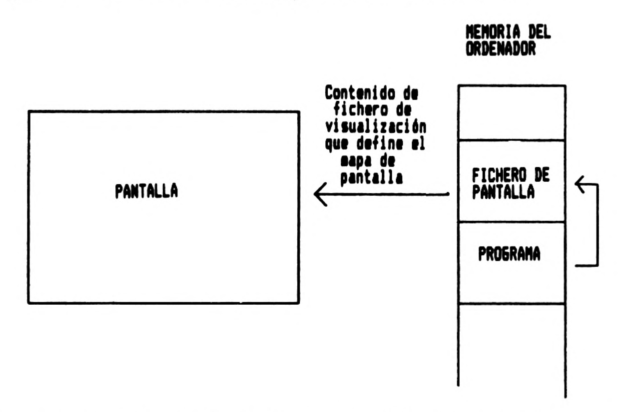


Figura 1.1 Relación entre la pantalla y la memoria del ordenador. El programa manipula las posiciones de la pantalla definidas en el mapa que estructura el fichero de pantalla, y la circuitería del ordenador refresca la imagen del fichero de pantalla 25 veces por segundo.

Recapitulemos sobre los términos introducidos hasta el momento. Los más importantes son: pantalla refrescada, fichero de pantalla, pixel y mapa de bits. Es preciso recordarlos, ya que aparecen con frecuencia en el campo de los gráficos.

En este punto quizá se esté Vd. preguntando cómo pasa la información desde el fichero de pantalla a la pantalla, o cómo está estructurado el propio fichero de pantalla. La respuesta a la primera pregunta es muy simple: olvídelo. Si realmente desea saber más acerca de los direccionamientos de la imagen y materias semejantes, necesitará un manual de *hardware*, y no este libro. Sin embargo, la estructura de la memoria de pantalla sí es interesante, pero es específica de cada máquina en gran medida.

No hemos concluido aún nuestra visión general de los conceptos gráficos. El siguiente término que introduciremos es la resolución de la pantalla gráfica. Seguramente ya estará Vd. familiarizado con los términos "gráficos en alta resolución" y en "baja resolución". En modo alta resolución, ésta viene definida por el número de pixels que pueden visualizarse en la pantalla. Una resolución de 1000 x 400 pixels implica que pueden visualizarse 400.000 puntos. Puede calcularse fácilmente el masivo fichero de pantalla que sería necesario para manejar tamaña resolución, incluso en monocromo: 400.000/8 bits por octeto (un bit por pixel), es decir, 50 Koctetos. ¡Está claro por qué los micros domésticos trabajan con resoluciones inferiores!

Los micros Amstrad disponen de una resolución máxima de pantalla de 640x200 pixels, lo que exige un fichero de pantalla de 16Koctetos. En efecto, como veremos a continuación, todos los modos de alta resolución del CPC6128,664 y 464 necesitan un espacio de 16Ko: la diferencia estriba en que algunos de estos modos ofrecen una paleta de colores más amplia, en detrimento de la resolución.

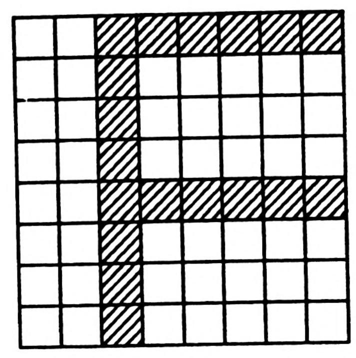


Figura 1,2 La salida de caracteres en "baja resolución" está constituida por una cuadrícula de 8 x 8 pixels. La configuración por puntos de cada carácter estándar está definida en una sección especial de la memoria a la que se accede cada vez que ha de imprimirse un carácter en pantalla.

Los gráficos en baja resolución están formados por bloques de pixels. Cada uno de estos bloques se considera como una unidad individual. La versión de estos bloques que utiliza el Amstrad mide 8x8 bits, o, lo que es lo mismo, 1 x 1 octetos *(bytes)*, como puede verse en el diagrama.

La principal aplicación de los gráficos en baja resolución son los textos, pero los gráficos de bloques emplean la misma técnica, y permiten al programador generar interesantes efectos gráficos utilizando únicamente la pantalla en baja resolución. En el Manual del Usuario podrá encontrar los caracteres para gráficos de bloques disponibles en el Amstrad. Existen dos técnicas de empleo de estos símbolos. En primer lugar, podemos "pegarlos" en la pantalla como un mosaico, para producir la imagen deseada, como puede ser el fondo de un juego de ordenador (Figura 1.3). También pueden utilizarse directamente para animación (por ejemplo, para los juegos de salón o de carreras).

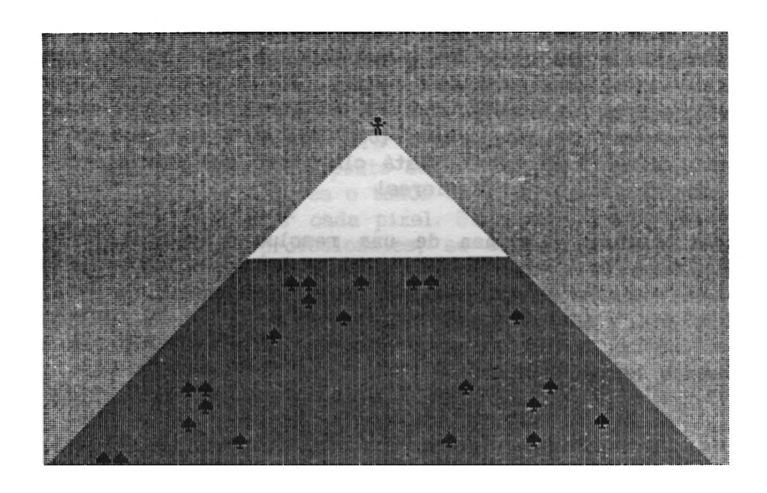


Figura 1.3 Ejemplo de imagen construida mediante celdillas de caracteres o "gráficos de bloque". Esta sencilla figura está compuesta mediante simples caracteres colocados en pantalla mediante la orden LOCATE.

La imagen de la Figura 1.3 ha sido producida utilizando solamente cinco caracteres gráficos de bloques distintos. Como puede comprobarse en el programa BLOQUE que se lista a continuación, todos ellos han sido dibujados empleando el comando PRINT CHR$. La orden LOCATE se utiliza para colocar en pantalla los símbolos gráficos en las filas y columnas adecuadas.

Programa BLOQUE

```
10 REM DEMOSTRACION DE GRAFICOS DE BLOQUE
20 REM PARA DIBUJAR UNA BONITA ESCENA EN MODO 1
30    MODE 1
40    INK 0,11:INK 1,0:INK 2,13:INK 3,9
50    CLS
60 REM PRIMERO DIBUJA LA MONTAÑA
70    PEN 3:REM LA MONTAÑA SE DIBUJA EN VERDE
80    J=26
90    FOR I=1 TO 19
100    IF J<15 THEN PEN 2
110    J=J-1
120    LOCATE I,J
130    PRINT CHR$(214)
140      FOR K=1 TO J-I+13
150        LOCATE I+K,J
160        IF J<14 THEN PEN 2
170        IF RND(1)<0.1 THEN GOSUB 300 ELSE PAPER 0:PRINT CHR$(143)
180      NEXT K
190      LOCATE I+K,J
200      PRINT CHR$(215)
210   NEXT I
220 REM AHORA DIBUJA EL MONTAÑERO
230    PEN 1
240    FIG=247
250    FIG=FIG+1:IF FIG=252 THEN FIG=248
260    LOCATE 20,6
270    PRINT CHR$(FIG)
280    FOR I=1 TO 500:NEXT I:GOTO 250
290 STOP
300 REM SUBRUTINA QUE DIBUJA LOS ARBOLES
310    IF K=1 OR K=J-I+13 THEN PAPER 0:PRINT CHR$(143):RETURN
320    IF J<14 THEN PEN 2:PRINT CHR$(143):RETURN
330    PEN 1:REM DIBUJA LOS OBJETOS EN NEGRO
340    PAPER 3:REM PONE DE COLOR VERDE EL FONDO DE LOS OBJETOS
350    PRINT CHR$(229)
360    PAPER 2
370    PEN 3
380 RETURN
```

Este programa es el primero y el último de este libro en el que utilizaremos los gráficos de bloques no de texto. Si lo que realmente desea es programar juegos sencillos mediante estos símbolos, ¡hay muchos más libros que le mostrarán cómo hacerlo!

1.3 Modos de Pantalla del Amstrad

Aunque la máxima resolución de pantalla del Amstrad es de 640 x 200 pixels, esta máquina puede trabajar en tres resoluciones diferentes. Estos modos (MODE 0, MODE 1 y MODE 2) proporcionan unas resoluciones efectivas de 160 x 200, 320 x 200 y 640 x 200 pixels, respectivamente. Además, en todas las modalidades la pantalla está definida como un área de 640 x 400 puntos, lo que implica que cada pareja de unidades a lo largo del eje Y (≡ vertical) estará representada en realidad por un solo pixel. Las unidades en el eje X son más flexibles. Trabajando en MODE 2, cada unidad equivale a un pixel. En MODO 1, un pixel contiene dos unidades, mientras que en MODE 0 cuatro unidades comparten un mismo pixel.

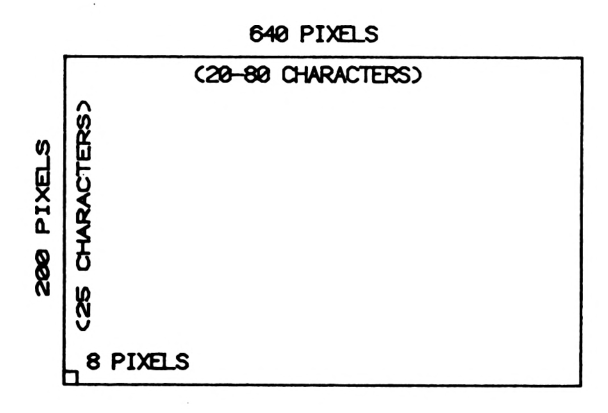


Figura 1.4 Comparación de las pantallas en alta y en baja resolución. Cada carácter se dibuja en realidad como una secuencia de 8 x 8 pixels.

Debemos hacer dos aclaraciones para resolver esta aparente confusión. En primer lugar, el empleo de un área de trabajo de 640 x 400 unidades proporciona la correcta relación de aspecto entre los ejes X e Y, y el hecho de que estas unidades sean las mismas en todos los modos simplifica enormemente la conmutación entre los distintos modos. La segunda consideración se refiere al empleo de los modos de más baja resolución, 1 y 0. ¿Qué razon hay para utilizarlos? La respuesta a esta pregunta radica en el uso del color. Si el verdadero número de pixels que contiene una memoria de pantalla de 16K es de 640 x 200, sólo habrá un bit para representar a cada pixel, con lo cual sólo podrán especificarse dos colores (primer plano y fondo). Si el número de pixels es de 320 x 200, podrán dedicarse a cada pixel dos bits de la memoria de pantalla, consiguiendo de esta forma cuatro posibles combinaciones de color. Una resolución de 160 x 200 añadirá dos bits más a la representación de cada pixel, obteniendo así 16 combinaciones de color en la pantalla al mismo tiempo.

Al ir avanzando en este libro podrá ver (y espero que utilizar) muchos programas distintos en los que intervienen los diversos modos de pantalla. Antes de dejar el tema de los modos, sería interesante considerar la visualización de los bloques de caracteres de texto en cada modo. Si consulta los Apéndices del manual de su Amstrad, podrá encontrar los distintos caracteres disponibles. Todos ellos se dibujan en una matriz de 8 x 8 puntos, como comentábamos anteriormente en este mismo capítulo. Si se escribe un carácter en MODE 0 ó 1, está claro que éste no ocupará un bloque cuadrado, como sucede en MODE 2. La resolución vertical (Y) en todos los modos es de 200 puntos, y la celdilla en que está contenido cada carácter tiene ocho unidades de altura. Dividiendo 200 entre 8 nos encontramos con que se dispone de 25 líneas de texto EN TODOS LOS MODOS. Veamos qué sucede con la resolución horizontal (X). En MODE 2 pueden visualizarse 640/8 = 80 caracteres. En MODE 1 pueden presentarse 320/8 = 40 caracteres, mientras que en MODE 0 sólo pueden escribirse 160/8 = 20 caracteres por línea.

En este libro nos ocuparemos principalmente de las aplicaciones gráficas escritas en BASIC, por lo que no es necesario preocuparse de la disposición del mapa de memoria del CPC 6128, CPC 664 ó CPC 464. Si realmente desea programar gráficos en código máquina, debe adquirir la Guía del *Firmware* del CPC 464 (AMSOFT 158), que puede solicitarse en Amstrad.

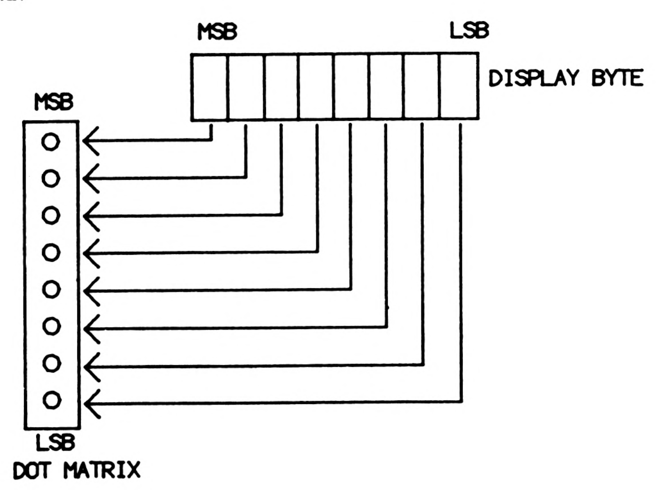


Figura 1.6 Un octeto dentro del fichero de pantalla podría volcarse a una impresora haciendo que las agujas de la cabeza de impresión copiasen el estado activado/desactivado de cada pixel del octeto. En las impresoras serie de Epson la cabeza de matriz de puntos es una banda de 8 x 1 agujas, por lo que cada octeto se transfiere directamente como una columna vertical de puntos. MSB = Bit más Significativo - *Most Significant Bit* (valor 128), LSB = bit menos significativo - *Least Significant Bit* (valor 1)

El fichero de pantalla de alta resolución no sólo proporciona un mapa de bits de la pantalla de vídeo; puede utilizarse también para entregar datos a otro dispositivo de visualización. Si se utiliza una impresora matricial, puede efectuarse un volcado de la pantalla para transferir al papel la información contenida en el fichero de pantalla, escribiendo secuencialmente cada bit de esta forma.

Un trazador gráfico *(plotter)* funciona según un principio bastante diferente, transfiriendo en conjunto todo el fichero de pantalla: dibuja líneas que unen puntos especificados directamente al trazador. La calidad de los gráficos generados puede ser muy elevada, puesto que cada línea es una auténtica recta, y no una secuencia de diminutos puntos. Cuatro de las cinco primeras figuras del libro han sido producidas por un trazador.

1.4 Tintas y colores

Seguramente el aspecto más confuso de la ejecución de programas en los ordenadores Amstrad es la elección del color. Como ya hemos visto, en MODE 2 se dispone de dos colores, de cuatro en MODE 1 y de un espectacular espectro de 16 colores en MODE 0. La "paleta" total de colores es de 27, todos los cuales pueden verse ejecutando el siguiente programa

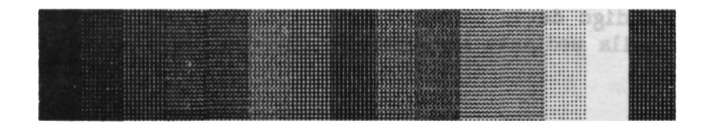
Programa COLOR

```
10 REM *** DEMOSTRACION DE COLORES ***
20 MODE 0
30 col1=0:col2=15
40 PAPER 0
50 CLS
60 IF col1=0 THEN LOCATE 1,2:PRINT"PALETA DE COLORES(1)"
70 IF col1=16 THEN LOCATE 1,2:PRINT"PALETA DE COLORES(2)"
80 m=-1
90 k=0:l=0
100 LOCATE 1,5
110 FOR j=col1 TO col2
120 m=m+1
130 INK m,j
140 NEXT j
150 FOR i=0 TO (col2-col1)*40
160 IF l>40 THEN l=1:k=k+1
170 l=l+1
180 GRAPHICS PEN k
190 MOVE i,0
200 DRAW i,300
```

```
210 NEXT i
220 IF col2=27 THEN 220
230 IF col1=0 THEN PRINT "PULSE UNA TECLA"
240 col1=16:col2=27:m=-1
250 INK 15,0
260 PAPER 15
270 a$=INKEY$
280 IF a$="" THEN 270
290 GOTO 50
```

PALETA DE COLORES 1

1 2 3 4 5 6 7 8 9 10 11 12 13 14



PALETA DE COLORES 2

15 16 17 18 19 20 21 22 23 24 25 26

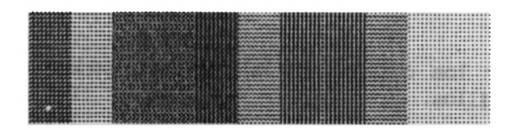


Figura 1,6 Salida del programa COLOR, El volcado de pantalla se ha hecho mediante el programa Tascopy, mostrando distintos tonos de gris para los diferentes colores. Observe que el modo 0 se emplea para conseguir 16 colores simultáneos en pantalla. Los números se añadieron después de imprimir (En MODE 0 los números son demasiado grandes)

Puede verse que algunos de los colores se parecen bastante. Ello se debe, sobre todo, a las limitaciones del monitor, más que a una mala elección de los colores por parte del programa en el Amstrad. Claro está que tal vez esté Vd. utilizando un monitor monocromo, en el cual los colores se reducen a una escala de grises. Pero no hay por qué preocuparse si no se dispone de un monitor en color. La mayoría de los programas de este libro, si no todos, seguirán siendo válidos.

Los colores del Amstrad se codifican mediante la orden INK, que se emplea de esta forma:

INK 0,1

donde el primer número representa el número del código de color para la tinta, y el segundo es el número del color del fondo, como se indica en el Manual del Usuario de su CPC 664/CPC 6128. Una especificación de los colores de esta forma sólo tendrá efecto inmediato si el código especificado para la tinta es 0 ó 1. El código de tinta 0 selecciona el color de la pantalla (por defecto), mientras que el código 1 ajusta el color por defecto de la tinta, con lo cual todos los caracteres y las líneas tomarán este color si no se indica lo contrario.

Puede cambiarse el código de la tinta de dibujo, el color del fondo o el del marco de la pantalla mediante los comandos

PEN n
PAPER n
BORDER n

donde n es el color de tinta escogido. La orden PAPER requiere una explicación más detallada. Sólo afecta al entorno de los caracteres de texto. Si se quiere cambiar el color de la pantalla, deberá usarse la orden

INK 0,n

donde n es el color deseado para la pantalla.

1.5 Algunos gráficos sencillos

Ahora que ya sabemos manipular los colores de la pantalla, podemos ver algunos programas gráficos sencillos.

Comenzaremos nuestras exploraciones gráficas con cuatro pequeños programas que nos harán tomarle gusto a la alta resolución. Los tres primeros, que dibujan un hexágono, un círculo y una espiral, utilizan las primitivas gráficas elementales - puntos y líneas - de las que nos ocuparemos en profundidad en el Capítulo 2. El último ejemplo, GRAFICAS, utiliza gráficos mezclados con texto.

Dibujemos en primer lugar un hexágono. Los datos del hexágono vienen dados en pares de coordenadas cartesianas (X,Y), como puede verse en las sentencias DATA. No se preocupe si el término "coordenadas" no le suena demasiado: los programas siguientes son en realidad muy simples, y tendremos tiempo más adelante de ocuparnos de la terminología.

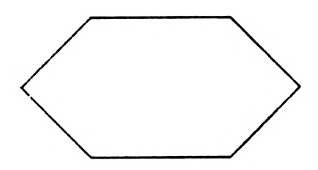


Figura 1,7 Resultado de HEXAGONO

Programa HEXAGONO

```
10 REM***PROGRAMA HEXAGONO***
20 REM CONSTRUYE UN HEXAGONO A PARTIR DE LOS DATOS COORDENADOS DE LA
SENTENCIA DATA
30 INK 0,0
40 INK 1,12
50 MODE 1
60 PAPER 0
70 GRAPHICS PEN 1:REM LAS LINEAS 30 - 70 DEFINEN LOS COLORES DE DIBUJO
80 READ X1,Y1
90 FOR I=1 TO 6
100 X=X1:Y=Y1
110 READ X1,Y1
120 MOVE X,Y
130 DRAW X1,Y1
140 NEXT I
150 END
160 DATA 100,150,200,150,250,100,200,50,100,50,50,100,100,150
```

Ya que este es el primer programa verdadero de "gráficos" de este libro, lo desglósaremos línea por línea, a pesar de que es ciertamente simple. Las líneas 10 y 20 son las consabidas sentencias REM. Las líneas 30 y 40

ajustan los colores del fondo (pantalla) y del primer plano (tinta). La línea 50 selecciona el modo 1, y la línea 60 confirma que el color de fondo del texto será el mismo que el de la pantalla. En la línea 80 se leen las coordenadas de comienzo del hexágono - en este caso se ha elegido la esquina superior izquierda, pero podría haberse escogido cualquiera de las otras esquinas -. Las coordenadas de comienzo se guerdan en las variables X1 e Y1, aunque esto es algo meramente temporal. En realidad es preciso que la coordenadas se guarden en las variables X e Y, como veremos más adelante en la línea 120. La línea 90 es el principio del bucle que dibuja las líneas. Este bucle se ejecuta seis veces, una para cada lado. La línea 100 asigna a las variables X e Y los valores de X1 e Y1, respectivamente. Ello se debe a que el final de una línea se convierte en el principio de la siguiente.

A continuación, en la línea 110, se leen las coordenadas del punto en que ha de terminar la siguiente línea a dibujar. Las líneas 120-130 son las que contienen las instrucciones para dibujar la línea propiamente dicha. La línea va desde X,Y hasta X1,Y1. La línea 140 marca el final del bucle, y la 160 contiene los datos de las coordenadas necesarias para dibujar el hexágono.

El programa HEXAGONO ilustra el empleo de algunas técnicas gráficas sencillas: el uso de las órdenes MOVE y DRAW, la selección de los colores de la pantalla, así como el manejo de datos gráficos rudimentarios. En HEXAGONO, todos los datos están definidos completamente por el programador. El próximo programa, CIRCULO, emplea una técnica diferente: aquí los datos están generados por una función matemática, en este caso la posición de los puntos a una distancia fija (radio) de un punto central.

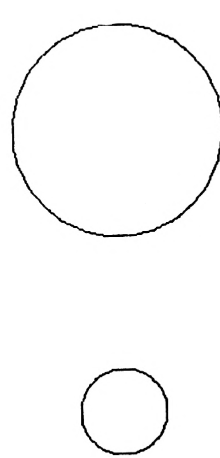


Figura 1,8 Resultado de CIRCULO

Aunque el programa dibuja aproximadamente un círculo, en realidad lo que produce es un polígono de 100 lados, calculando secuencialmente las posiciones de 100 puntos alrededor del centro. Es una técnica algo chapucera y lenta, por lo que sólo sirve de demostración: no deja de ser curioso que el BASIC de Amstrad no incluya una orden CIRCLE, bastante habitual en muchas otras versiones de BASIC que funcionan en micros distintos. Es un inconveniente insignificante en una máquina por otra parte excelente (que además nos sirve de excusa para jugar con algoritmos de dibujo de circunferencias).

Programa CIRCULO

```
10 REM****PROGRAMA CIRCULO****
20 REM CONSTRUYE UN CIRCULO A PARTIR DE COORDENADAS CALCULADAS
30 INK 0,0
40 INK 1,12
50 MODE 1
60 PAPER 0
70 GRAPHICS PEN 1:REM LAS LINEAS 30 - 70 DEFINEN LOS COLORES DE DIBUJO
80 INPUT "RADIO";R
90 AN=0
100 AI=0.062831853
110 X1=R*COS(AN):Y1=R*SIN(AN)
120 FOR I=1 TO 100
130 X=X1:Y=Y1
140 AN=AN+AI
150 X1=R*COS(AN):Y1=R*SIN(AN)
160 MOVE X+320,Y+200
170 DRAW X1+320,Y1+200
180 NEXT I
190 END
```

El siguiente programa utiliza también una función matemática para generar un trazado, que en este caso es una espiral. ESPIRAL no hace muchas más cosas que CIRCULO, pero puede producir resultados realmente estéticos si se asigna a RE (la resolución) un valor lo bastante grande.

Programa Espiral

```
10 REM****PROGRAMA ESPIRAL****
20 REM CONSTRUYE UNA ESPIRAL A PARTIR DE COORDENADAS CALCULADAS
30 INK 0,0
40 INK 1,12
50 MODE 1
60 PAPER 0
```

```
70 GRAPHICS PEN 1:REM LAS LINEAS 30 - 170 DEFINEN LOS COLORES DE DIBUJO
80 INPUT"RESOLUCION";RE
90 X1=320:Y1=200
100 FOR I=0 TO 50 STEP RE
110 R=I*2:X=X1:Y=Y1
120 X1=R*SIN(I)+320:Y1=R*COS(I)+200
130 MOVE X,Y
140 DRAW X1,Y1
150 NEXT I
```

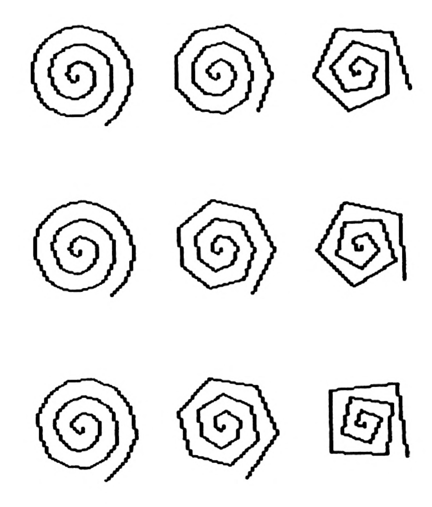


Figura 1,9 Resultado de ESPIRAL, Se han empleado nueve valores distintos de RE,

El siguiente programa utiliza las primitivas gráficas elementales para crear una gráfica. Las escalas de la misma, las leyendas que aparecen en los ejes y las coordenadas de los puntos se introducen durante la ejecución del programa. Este programa puede mejorarse fácilmente para que lea los datos de un fichero de disco o cinta, o para que visualice más de un conjunto de datos. (El acceso a ficheros secuenciales de datos se trata en el Capítulo 3).

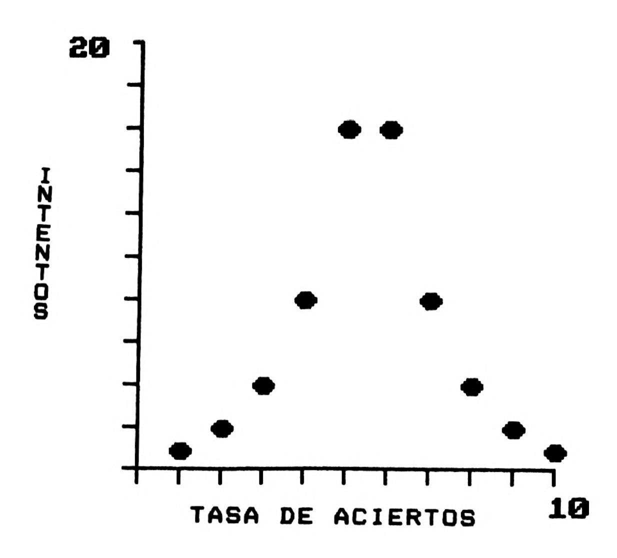


Figura 1.10 Resultado de GRAFICA

Programa GRAFICA

```
10 REM****PROGRAMA GRAFICAS****
20 REM TRAZA UNA SENCILLA GRAFICA ETIQUETADA
30    INK 0,13
40    INK 1,0
50   MODE 1
60    INPUT"CUANTOS PUNTOS EN LA GRAFICA";P
70 DIM X(P),Y(P)
80    FOR I=1 TO P
90    INPUT"VAL-X";X(I)
100   INPUT"VAL-Y";Y(I)
110   NEXT I
120   INPUT"CUAL ES EL MAXIMO VALOR EN EL EJE X";MX
130   INPUT"CUAL ES EL MAXIMO VALOR EN EL EJE Y";MY
140   INPUT"NOMBRE EJE X";N$
150   INPUT"NOMBRE EJE Y";M$
160 CLS
170 REM DIBUJAR LOS EJES
180   MOVE 200,380
190   DRAW 200,80
200   DRAW 500,80
210 REM TRAZAR LAS MARCAS DE LA ESCALA
220   FOR I=1 TO 11
230     MOVE 190,(I*30)+50
240     DRAW 200,(I*30)+50
```

```
250    NEXT I
260    FOR I=1 TO 11
270      MOVE (I*30)+170,70
280      DRAW (I*30)+170,80
290    NEXT I
300 REM ETIQUETADO DE LOS EJES
310 REM PRIMERO LA ETIQUETA DE LAS X
320 REM LA POSICION DE COMIENZO ES EL PUNTO CENTRAL DEL EJE X MENOS LA
MITAD DE
321 REM    LA LONGITUD DE LA CADENA
330    AX=(350-((LEN(N$)*16)/2))
340 REM LA POSICION DE COMIENZO ES EL PUNTO CENTRAL DEL EJE Y MAS LA
MITAD DE
341 REM LA LONGITUD DE LA CADENA
350    AY=(240+((LEN(M$)*16)/2))
360 TAG
370       MOVE AX,50
380    PRINT N$;
390    IF INKEY$="" THEN 390
400 REM AHORA IMPRIME LA LEYENDA DEL EJE Y EN VERTICAL
410    FOR I=1 TO LEN(M$):M1$=MID$(M$,I,1)
420      MOVE 120,AY-((I-1)*16)
430      PRINT M1$;
440    NEXT I
450    MOVE 480,60:PRINT MX;
460    MOVE 130,382:PRINT MY;
470 REM AHORA DIBUJA LOS PUNTOS
480    FOR I=1 TO P
490      MOVE 194+(300*(X(I)/MX)),86+(300*(Y(1)/MY))
500      PRINT CHR$(231);
510    NEXT I
```

GRAFICA es un programa algo más complejo que los considerados hasta ahora. Estas son sus principales secciones:

LINEAS	10-50	TITULO DEL PROGRAMA, SELECCION DE COLORES, MODO
	60	ENTRADA DEL NÚMERO DE PUNTOS DE LA GRAFICA
	70	DEFINICIÓN DE LAS MATRICES DE DATOS
	80	ENTRADA DE LAS COORDENADAS DE LOS PUNTOS
	120-150	ENTRADA DE LAS LEYENDAS Y ESCALAS DE LOS EJES
	155	BORRADO DE LA PANTALLA
	160-280	TRAZADO DE LOS EJES
	290-414	ETIQUETADO DE LOS EJES
	420-450	DIBUJO DE LOS PUNTOS

† En castellano la palabra "matriz" se refiere tanto a las estructuras de datos del lenguaje BASIC (ver Capítulo 3) como a las herramientas matemáticas de manipulación de vectores (ver Apéndice 2). La base de ambos conceptos es muy similar, por lo que habrá de ser el contexto el que nos indique a qué "matrices" nos estamos refiriendo. En este caso, se trata de las matrices de BASIC. (Nota del Traductor)

Por primera vez, este programa introduce la idea de manejar *matrices* para guardar datos. Tendremos ocasión de comprobar que esta técnica es uno los métodos básicos de los gráficos por ordenador. En el Capítulo 3 nos ocuparemos con mayor detalle de las matrices de BASIC. El programa GRAFICA ilustra también el empleo de las variables en un simple bucle FOR/NEXT para dibujar una serie de líneas. Se trata de una valiosa técnica cuyo funcionamiento es el siguiente: en el programa GRAFICA se desea dibujar una serie de marcas a lo largo de cada eje. Estas marcas podrían dibujarse laboriosamente mediante una serie de instrucciones MOVE y DRAW, pero ¿por qué hacer todo ese trabajo si el ordenador puede hacerlo por nosotros? Observemos las líneas 210 a 240 de GRAFICA, que controlan las marcas en el eje Y. La línea 170 establece el número de marcas que han de dibujarse: cualquier número es válido, dentro de los límites de resolución del ordenador. Las líneas 220-230 contienen las instrucciones para dibujar cada marca. Las coordenadas X de comienzo y de final son 190 y 200, respectivamente, pero en lugar de dibujar la misma marca varias veces debemos movernos hacia abajo (o hacia arriba) por el eje Y para cada nueva marca. Las coordenadas Y iniciales y finales serán la misma para cada marca del eje Y. Las parejas sucesivas de coordenadas Y van tomando valores múltiplos del contador I del FOR/NEXT: en este caso el multiplicador es 30. Como las marcas deben colocarse en Y = 50,80,110, y así sucesivamente, en lugar de en 0,30,60,..., debe sumarse a cada valor de Y un valor constante igual a 50. Para el eje X se utiliza la misma técnica en las líneas 250 a 280 de GRAFICA.

1.6 Colocación de texto

Existen dos formas de colocar texto en la pantalla de alta resolución. Como las pantallas de texto y de gráficos de los ordenadores Amstrad comparten la misma zona de memoria, podemos acceder a ambas al mismo tiempo. Esto significa que se puede utilizar la orden habitual de BASIC

```
PRINT "LO QUE QUIERAS"
```

para añadir texto a una imagen. El problema es que una orden tan escueta como ésta situará el texto en la posición actual del cursor, que casi con toda seguridad no será la adecuada. Para solucionar este problema, el BASIC de Amstrad ofrece la orden LOCATE. En el caso más sencillo, por ejemplo, la sentencia

```
LOCATE 10,20
```

sitúa el cursor en la décima columna, vigésima fila.

Desgraciadamente, el uso de LOCATE tiene ciertas limitaciones en alta resolución. Para poder utilizar LOCATE para etiquetar una figura es preciso transformar la posición de comienzo, fila y columna (direccionada dentro de un área de 25 x 40 en MODE 1) a la escala de unidades de 640 x 200. Esto puede hacerse, por supuesto, ya que el tamaño de la celdilla de cada carácter es constante. Sin embargo, exige unos cálculos innecesarios.

La solución es emplear una nueva orden llamada TAG, que permite que la posición de comienzo del cursor sea un pixel determinado, así

```
TAG
MOVE 100,200
PRINT"LO QUE QUIERAS";
TAGOFF
```

imprimirá la cadena de caracteres empezando en el pixel 100,200. Obsérvese que la orden TAG es algo peligrosa, ya que todo texto será "atado" a la posición actual del cursor gráfico hasta que se use TAGOFF. Conviene advertir (aunque quizá ya lo sepa) que tanto la orden LOCATE como TAG pueden especificarse para direcciones de salida determinadas. De ellas nos ocuparemos a continuación en la sección 7.

1.7 Impresión de gráficos

Aunque el principal objetivo de todo trabajo gráfico es producir la salida esperada del programa por pantalla, el obtener una copia impresa, ya sea generada por una impresora matricial o por un trazador de plumillas, es algo útil y recomendable. ¿De qué opciones dispone el propietario de un Amstrad? Amstrad fabrica una impresora, la DMP 1, que puede utilizarse para imprimir texto y gráficos.

La DMP 1 es una de las impresoras más baratas del mercado; su precio en Inglaterra es de unas 200 libras (Agosto de 1985). Puede imprimir todo el juego de gráficos de baja resolución del Amstrad: es una opción útil si se desea utilizar las capacidades gráficas de baja resolución de la máquina. La calidad de impresión que ofrece la DMP 1 es ciertamente baja, y sólo puede usar papel perforado, ya que únicamente dispone de mecanismo de tracción, y no de fricción para papel normal. La DMP 1 al menos permite el "volcado gráfico" (es decir, imprimir todos los pixels de la pantalla).

Pero lo bueno de los ordenadores Amstrad es que poseen una interfaz paralelo Centronics estándar. Esto significa que cualquier impresora compatible Centronics puede conectarse con sólo comprar el cable adecuado (observe que el enchufe de la parte posterior del ordenador no es un conector tipo Centronics, por lo que deberá adquirir un cable especial, por unas 4.000 pesetas).

Especificando el canal 8 en los comandos de salida, la salida puede escribirse directamente en la impresora. El volcado de la pantalla no puede realizarse directamente: es necesario un programa adecuado que realice esta tarea. Los últimos números de algunas revistas especializadas incluyen listados de programas para volcar la pantalla. Existen tres tipos de programas de este género. El peor de todos ellos es el de los programas de volcado escritos en BASIC - ¡pueden tardar hasta media hora en volcar una pantalla! Los listados de revistas que emplean código

máquina son mucho más rápidos, aunque a veces no permiten sombreados,y quizá no puedan copiar toda la anchura de la pantalla.

La mejor solución consiste en comprar uno de los paquetes comerciales de *software* disponibles en el mercado. Este tipo de programas suele ser capaz de manejar toda la anchura de la pantalla, imprimiendo la misma en vertical (el eje más largo en sentido longitudinal con respecto al papel). El mejor paquete es, seguramente, Tascopy (de Tasman Software Ltd). Tascopy permite producir copias de la pantalla en todos los modos, representando los colores en una completa escala de grises (ver,por ejemplo, las Figuras 1.6 y 1.11).

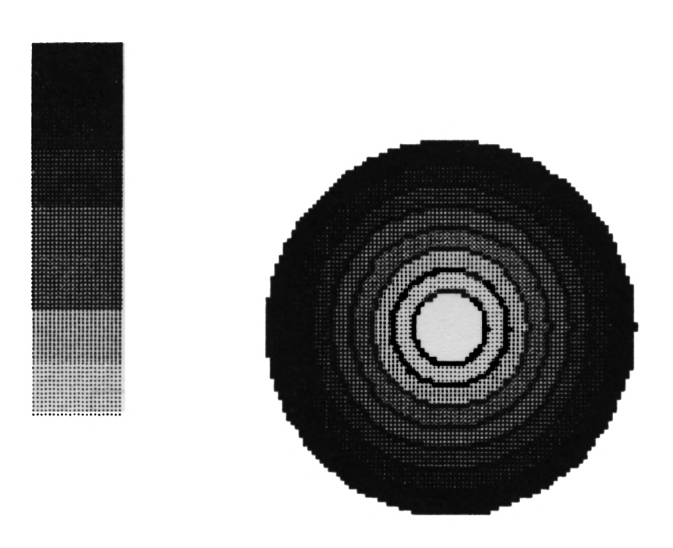


Figura 1.11 Ejemplo de sombreado de colores con el programa Tascopy

Los tonos de grises pueden ser especificados por el usuario. Tascopy permite además producir copias de la pantalla en tamaño gigante, en cuatro hojas de papel de la impresora (dibujando una cuarta parte de la pantalla en cada hoja, que ha de ser pegada junto con las otras). El paquete permite utilizar una amplia variedad de impresoras, entre ellas la DMP 1. Todos los volcados de pantalla que aparecen en este libro han sido realizados con Tascopy.

Suponiendo que dispone de una impresora capaz de copiar la pantalla de alta resolución, quizá advierta que las proporciones de los ejes X e Y en la impresora son diferentes de las de la pantalla. Para comprobarlo, intente imprimir la salida del programa CIRCULO. El método de ensayo y error le mostrará cómo compensar esta diferencia. El empleo de un paquete sofisticado como Tascopy obviará este problema.

El siguiente programa (PANTALLA) traza las dimensiones de la pantalla, y puede utilizarse para comprobar el tamaño del volcado de la misma y la relación XY disponible en nuestra impresora. Por supuesto, habrá que incluir el comando de volcado de pantalla apropiado para el programa de impresora que se esté utilizando.

Programa PANTALLA

```
10 REM****PROGRAMA PANTALLA****
20 CLS
30 MOVE 1,1
40 DRAW 639,1
50 DRAW 639,399
60 DRAW 1,399
70 DRAW 1,1
80 REM  AHORA DIBUJA UNA CRUZ EN EL MEDIO DE LA PANTALLA
90 MOVE 310,200
100 DRAW 330,200
110 MOVE 320,190
120 DRAW 320,210
```

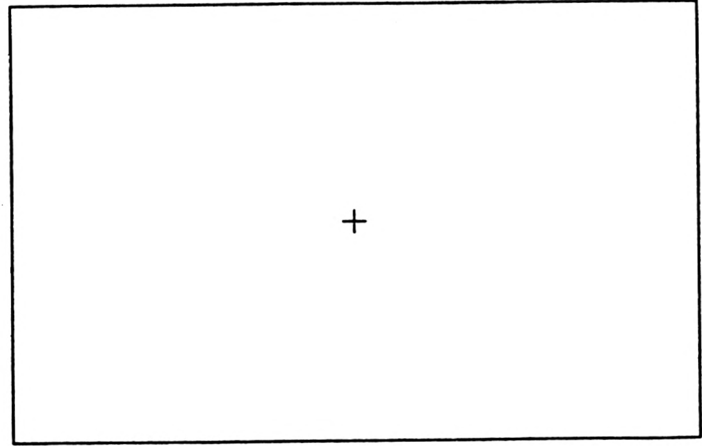


Figura 1.12 Resultado de PANTALLA

En este capítulo hemos mostrado el uso de los ordenadores Amstrad para generar imágenes gráficas, pero no nos hemos ocupado aún de una forma lógica de las órdenes empleadas para crear los gráficos en la pantalla. Investigaremos estas órdenes en el Capítulo 2.

Capítulo 2

Puntos, líneas y contornos

2.1 Cómo dibujar líneas

Ya hemos visto en el Capítulo 1 algunos programas gráficos sencillos, pero aún no hemos explicado las diversas órdenes gráficas de que disponemos en el Amstrad. En el Apéndice I aparece una lista de las órdenes gráficas utilizadas en el CPC 6128, 664 y 464. En este capítulo configuraremos todo nuestro arsenal de órdenes con los comandos gráficos que se utilizarán con mayor frecuencia a lo largo del texto. Comencemos con las "primitivas gráficas" más sencillas: los puntos y las líneas.

Para empezar no nos ocuparemos de momento de los canales ni de los colores. Supongamos que se ha de dibujar una línea desde el punto X1,Y1 hasta el punto X2,Y2. Para hacer esto en un ordenador Amstrad deben utilizarse las siguientes sentencias:

```
MOVE X1,Y1
DRAW X2,Y2
```

La posición actual del "cursor de gráficos" se desplazará hasta el punto especificado por la última orden gráfica. Después de las dos sentencias anteriores quedará, pues, en la posición X2,Y2. Recordemos que las instrucciones MOVE y DRAW ya fueron utilizadas en el Capítulo 1.

Al comenzar la sesión de programación gráfica, el cursor de gráficos estará en la posición 0,0 - esquina inferior izquierda de la pantalla. Este punto se conoce como "origen". El BASIC de Amstrad dispone de un comando adicional llamado, de forma muy apropiada, ORIGIN, que permite desplazarse al origen desde cualquier punto del área de pantalla. La línea dibujada anteriormente puede trazarse ahora mediante la orden ORIGIN

```
ORIGIN X1,Y1
DRAW X2,Y2
```

El empleo de ORIGIN permite especificar coordenadas *negativas*. Si intenta trazar la siguiente línea

```
DRAW -10,-10
```

obtendrá un sólo punto en la posición 0,0, ya que todos los demás puntos desde 0,0 hasta -10,-10 están fuera de la pantalla. En cambio, si se utiliza

DRAW -10,-10 ORIGIN -100,-100

se dibujará, en efecto una línea, ya que el verdadero punto final de la línea está en la posición 90,90.

Ya hemos visto cómo dibujar líneas, pero a veces es necesario también borrarlas, ya sea porque se han dibujado en un lugar erróneo, o porque la imagen debe ser actualizada (girada, por ejemplo). La forma más habitual de hacerlo es incluir un parámetro adicional en la orden DRAW, que definirá lo que debe hacerse con cada uno de los pixels que atraviese la línea. En el Manual de Usuario encontrará este parámetro como "tinta"; se dispone de cuatro de estas tintas, a saber,

0, Dibuja normalmente
1 Efectúa una operación lógica OR-exclusivo (XOR) con cada pixel
2 Efectúa una operación AND con cada pixel
3 Efectúa una operación OR con cada pixel

Si no se especifica este parámetro, el ordenador asume el valor por defecto 0. El uso del modo de tinta 1 es especialmente útil, ya que permite "borrar" pixels encendidos.

El siguiente programa nos muestra el uso de esta capacidad de borrado, dibujando líneas aleatoriamente en la pantalla y borrándolas de nuevo inmediatamente.

Programa AHORALOVES

```
10 REM****PROGRAMA AHORALOVES****
20 RANDOMIZE
30 CLS
40 X1=RND(1)*640:REM  define el primer valor de la X
50 Y1=RND(1)*400:REM  define el primer valor de la Y
60 X2=RND(1)*640:REM  define el segundo valor de la X
70 Y2=RND(1)*400:REM  define el segundo valor de la Y
80 MOVE X1,Y1
90 DRAW X2,Y2
100 REM  ahora borra la linea
110 REM  en el modo de tinta 1 (XOR) se borrara la linea existente
120 MOVE x1,y1,1,1
130 DRAW x2,y2,1,1
140 GOTO 40
```

Especificando el modo de tinta 1 pueden "invertirse" los pixels (es decir, apagarlos si están encendidos y encenderlos si están apagados). La inversión tiene dos aplicaciones principales. La primera y menos seria de ellas es la creación de preciosos efectos gráficos. El programa INVERSION que se ofrece a continuación nos muestra la posibilidad de producir complejos e impresionantes resultados gráficos con muy poco esfuerzo de programación.

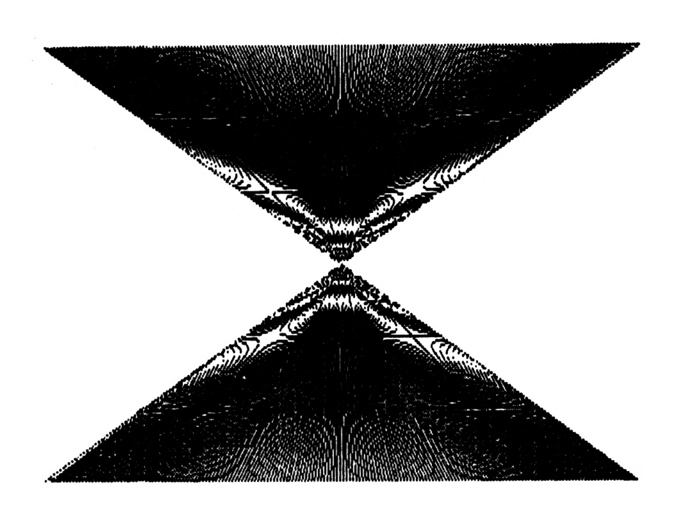


Figura 2.1 Resultado de INVERSION

Programa INVERSION

```
10 REM***PROGRAMA INVERSION***
15 CLS
17 n=0
20 x1=40:y1=0
30 x2=600:y2=400
40 x1=x1+3:x2=x2-3
50 IF x1>600 THEN x1=41:n=n+1
60 IF x2<40 THEN x2=599
70 MOVE x1,y1,1,1
80 DRAW x2,y2,1,1
83 IF n=1 THEN n=2
85 IF x1=41 THEN INK 1,n
90 GOTO 40
```

La aplicación más práctica de la técnica de inversión es evitar el borrado de elementos de la imagen cuando se escribe sobre parte de la misma. Utilizaremos la inversión en el Capítulo 3 para prevenir el borrado de una imagen al mover el cursor por encima de ella. Consideremos una línea vertical que atraviesa por el medio la pantalla. Si movemos un objeto por la pantalla, borrándolo, por ejemplo, de la posición X1, y escribiéndolo de nuevo en la posición X2, entonces, cuando el objeto atraviese la línea vertical, la orden de borrado hará desaparecer todos los puntos de la línea cuya posición coincida con la del objeto a borrar. Si en lugar del borrado utilizamos la inversión, el objeto seguirá desapareciendo cuando así se indique (ya que si todos los pixels encendidos se invierten, quedarán apagados). ¿Cómo afecta esto a la línea vertical? Cuando el objeto alcanza la línea, la inversión apaga los pixels de la misma que coincidan con el objeto, y cuando la posición de éste vuelve a invertirse (para hacerlo desaparecer), la línea vertical se reconstruye. En la Figura 2.2 podemos ver gráficamente este proceso.

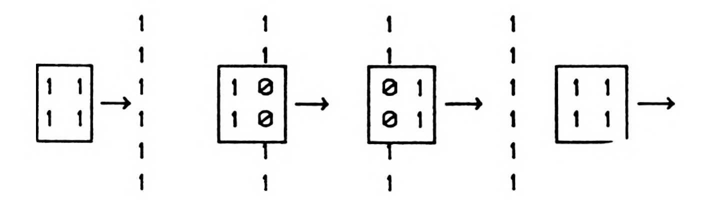


Figura 2.2 La técnica de inversión en funcionamiento. La secuencia de imágenes muestra el movimiento de un objeto cuadrado que cruza una línea vertical, sin borrar de forma permanente ningún pixel. Cuando se encuentran los pixels del cuadrado con los de la línea, se invierten, con lo cual se apagan (es decir, 1 + 1 = 0, en binario). Cuando el cuadrado abandona la línea vertical, al invertirse de nuevo los pixels que fueron desactivados por el cruce con el cuadrado, vuelven a encenderse.

2.2 Puntos

Hasta ahora hemos considerado únicamente líneas, pero suele ser necesario dibujar pixels individuales. El BASIC de Amstrad utiliza para ello la orden PLOT, cuyo manejo es muy parecido al de DRAW. Para colocar un punto individual en la posición X1,Y1,

```
PLOT X1,Y1
```

y, para dibujar una línea de puntos horizontal que atraviese la pantalla,

```
MOVE 100,100
      FOR X=100 TO 600 STEP 10
      PLOT X,100
      NEXT X
```

La generación de puntos en lugar de líneas puede ser interesante si se desea saber si dos líneas se cortan. Puede hacer falta dibujar, por ejemplo, una línea que no corte a ninguna otra. Pruebe con este pequeño programa:

Programa CRUCE

```
10 REM *** programa cruce ***
20 REM *** demuestra el uso del comando test ***
30 CLS
40 INK 0,13:INK 1,0
50      MOVE 300,300
60      DRAW 300,100
70        FOR x=0 TO 640
80        IF TEST(x,200)<>0 THEN 200
90        PLOT x,200
100       NEXT x
200 STOP
```

Este programa dibuja una línea vertical, y construye a continuación una línea horizontal dibujando los pixels adyacentes. La orden TEST(X,Y) comprueba si el pixel de la posición X,Y que ha de encenderse para continuar la línea horizontal está ya encendido. Cuando la línea horizontal alcanza la vertical, esta condición se cumple, con lo cual el programa se detiene.

2.3 Cómo dibujar contornos

Vimos en el capítulo anterior que mediante el uso reiterado de órdenes DRAW podemos dibujar contornos. Un experimento interesante en este aspecto puede consistir en dibujar un diseño sobre un papel milimetrado de dimensiones 640 * 400 unidades. Para hacerlo en el ordenador basta especificar una orden DRAW para cada punto adyacente del contorno. Recuerde que debe utilizar una orden MOVE para desplazar el cursor al primer punto, o aparecerá una fea línea adicional desde el origen hasta ese punto. Si algunas partes del diseño están aisladas del resto, serán necesarias algunas órdenes MOVE más.

2.4 Trazos y rellenos

El afortunado propietario de un CPC 6128 ó CPC 664 dispone en su arsenal de comandos gráficos de algunas órdenes en este sentido que no están presentes en el CPC 464. En concreto, el comando MASK permite dibujar líneas discontinuas de diversos aspectos, en lugar de una línea continua. Esta orden tiene el siguiente formato:

MASK numero

donde "número" es un valor entero comprendido entre 0 y 255. MASK activa o desactiva cada uno de los puntos de los grupos sucesivos de ocho pixels. El siguiente programa visualiza algunos patrones de trazos típicos (recuerde que esto sólo funciona en el CPC 664/6128).

EJEMPLOS DE VALORES DE MASK

MODO = 1

1
3
33
7
15
31
63
127

Figura 2,3 Resultado de DEMOMASK

Programa DEMOMASK

```
10 REM *** programa de demostracion de MASK ***
15 MODE 1
20 CLS
25 LOCATE 5,2:PRINT "ejemplos de valores de MASK"
27 LOCATE 5,4:PRINT "modo = 1"
30 y=300
40 DATA 1,3,33,7,15,31,63,127
50 FOR i=1 TO 8
60   READ m
70   MASK m
80  y=y-30
90   MOVE 100,y
100  DRAW 400,y
110  TAG
120 PRINT m;
140 NEXT i
```

Para utilizar eficazmente la orden MASK hemos de recordar (o aprender, qué vergüenza) algunos conceptos elementales sobre numeración binaria. Recordemos que el número 255 se representa por

1 1 1 1 1 1 1 1

y que el 0 binario se representa como

0 0 0 0 0 0 0 0

así, si especificamos una "máscara" (MASK) de 255, se generará una línea continua, mientras que una máscara 0 no producirá línea alguna. Si se desea dibujar un punto sí y otro no, la máscara a elegir será

1 0 1 0 1 0 1 0

o bien

0 1 0 1 0 1 0 1

que corresponden a los números decimales 170 y 85, respectivamente.

Si no dispone de un CPC 6128/664, no todo está perdido. Con el siguiente programa podrá dibujar una línea que una dos puntos utilizando una máscara de trazos.

Programa TRAZOS

```
10 REM  para usuarios de cpc 464 sin orden MASK
20 REM  dibuja lineas de trazos desde x1,y1 hasta x2,y2
30     CLS
40     INPUT"incremento";incremento
50     GOSUB 240: REM crea los puntos finales
60     MOVE x1,y1
70 REM calcula en primer lugar la longitud de la linea
80     hy2=(x2-x1)↑2+(y2-y1)↑2
90     hy=SQR(hy2)
100       inc=incremento*(hy/300)
110       PLOT x1,y1,1,0
120       PLOT x2,y2,1,0
130 REM ahora halla las relaciones para los incrementos de x y de y
140     xr=(x2-x1)/inc:yr=(y2-y1)/inc
150 REM ahora dibuja la linea
160     trazo=0
170       FOR i=1 TO inc
180         IF trazo=0 THEN trazo=2:GOTO 200
190         IF trazo=2 THEN trazo=0
200         x1=x1+xr:y1=y1+yr
210         DRAW x1,y1,1,trazo
220       NEXT i
230     GOSUB 240:GOTO 60
240 REM rutina para un punto final aleatorio
250     x1=RND(1)*600:x2=RND(1)*600
260     y1=RND(1)*400:y2=RND(1)*400
270 RETURN
```

```
INCREMENTO? 20
```

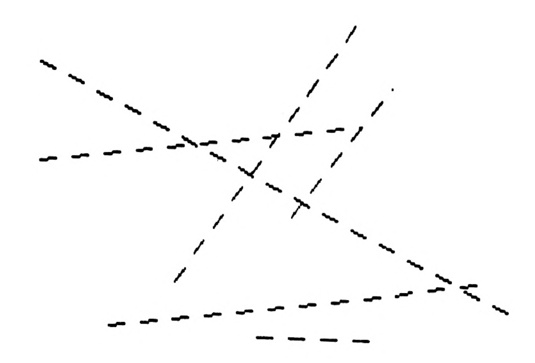
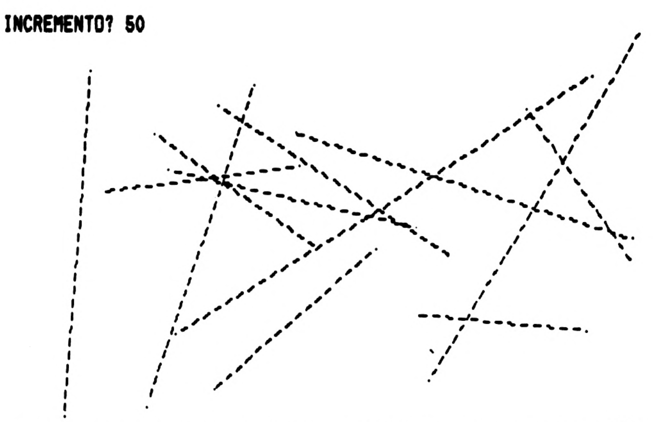


Figura 2.4



Figuras 2.4, 2.5 Dos ejemplos de figuras creadas por el programa TRAZOS

Quizá la limitación más grave del CPC 464 es la falta de una orden FILL. FILL rellenará un recinto DESDE LA POSICIóN ACTUAL DEL CURSOR DE GRAFICOS con el color elegido. Su sintaxis es muy simple:

FILL numero de tinta

La operación de rellenado sólo se detiene cuando se alcanzan pixels encendidos, por lo que, si se intentara rellenar un recinto que no estuviese limitado por un contorno de pixels encendidos y adyacentes, podría rellenarse toda la pantalla. No puedo ofrecer en este libro una simulación completa de FILL para el CPC 464, pero el programa llamado PATRóN, que aparece en el Capítulo 5, permite rellenar áreas rectangulares (por ejemplo, en gráficos de barras) con diversos patrones de sombreado.

2.5 Cómo dibujar curvas

En el Capítulo 1 veíamos cómo dibujar un círculo. En muchas ocasiones es necesario construir otros contornos curvilíneos. En primer lugar, consideremos una elipse. Las ecuaciones que utilizaremos serán las mismas que las empleadas para el círculo, pero en este caso emplearemos radios diferentes para los ejes X e Y. Pruebe esta versión del programa CIRCLE (al que llamaremos ahora ELIPSE) para diferentes valores de RX y RY.

Programa ELIPSE

```
10 REM ***PROGRAMA ELIPSE***
20 REM construye una elipse a partir de coordenadas calculadas
30 INK 0,0
40 INK 1,12
50 MODE 1
60 PAPER 0
70 GRAPHICS PEN 1:REM las lineas 30 - 70 definen los colores de dibujo
80 INPUT"radios X,Y";xr,yr
90 an=0
100 ai=0.062831853
110 x1=xr*COS(an):y1=yr*SIN(an)
120 FOR i=1 TO 100
130   x=x1:y=y1
140   an=an+ai
150   x1=xr*COS(an):y1=yr*SIN(an)
160   MOVE x+320,y+200
170   DRAW x1+320,y1+200
180 NEXT i
190 END
```

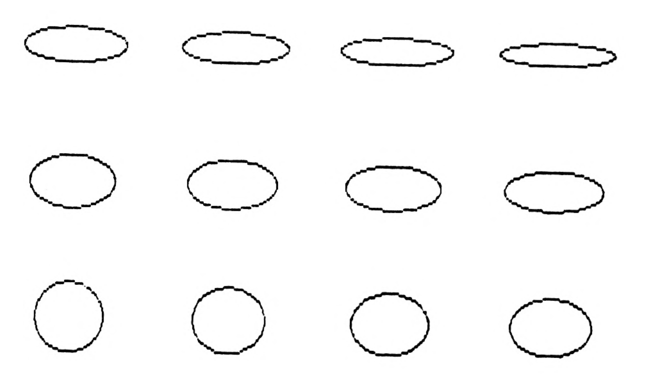


Figura 2.6 Elipses con diversos valores de RX y RY dibujadas mediante el programa ELIPSE

La ecuación general de una curva sinusoidal puede escribirse como

Y = H * SEN (W * X + D)

donde W es la frecuencia (que determina el número de oscilaciones) para un determinado margen de valores de X. D especifica el desplazamiento de la curva hacia la derecha (positivo) o hacia la izquierda (negativo), también conocido como desfase. He aquí un programa que genera curvas sinusoidales

Programa SENO

```
10 REM ***PROGRAMA SENO***
20 CLS
30 INK 0,13:INK 1,0
40     INPUT"valor maximo de la Y";yv
45     m=yv/2
50     INPUT "alt,an,prof";h,w,d:REM utilice como prueba los valores 40,.1,0
60     IF h>m THEN 40
70     INPUT"min(x),max(x)";xmin,xmax
80      ya=h*SIN(w*xmin+d)
90      IF ya>=0 THEN ya=m-ya
100     IF ya<0 THEN ya=m+ABS(ya)
110     xa=xmin
120       FOR xb=xmin TO xmax
130         yb=h*SIN(w*xb+d)
140         IF yb>=0 THEN yb=m-yb
150         IF yb<0 THEN yb=m+ABS(yb)
160         MOVE xa,ya
170         DRAW xb,yb
180         xa=xb
190         ya=yb
200       NEXT xb
210 END
```

CURVA SENO: H=175,W=.05,D=0

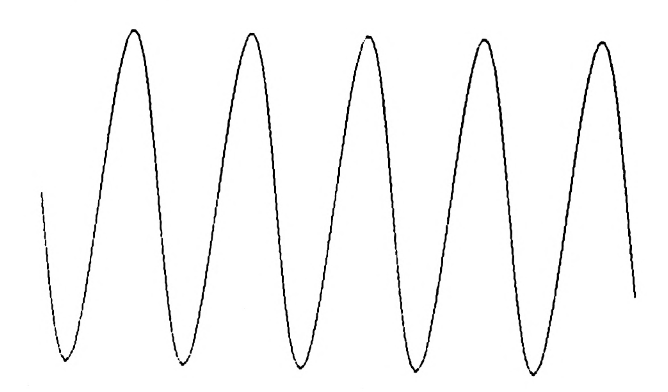


Figura 2,7   Curva sinusoidal dibujada mediante el programa SENO

Una parábola es una curva muy útil que puede emplearse para describir el movimiento de los objetos. La ecuación para calcular una coordenada Y es

$$Y = C1 * X^2 + C2 * X + C3$$

donde C1, C2 y C3 son tres coeficientes que pueden cambiarse para producir distintas trayectorias parabólicas. El programa PARA dibuja una curva parabólica una vez introducidos los valores máximos de X e Y y los valores de C1, C2 y C3. Intente modificar esos coeficientes. La parábola tomará valor máximo en el centro si C1 > 0.

Programa PARABOLA

```
10 REM ***PARABOLA***
20 REM dibuja una parabola
25 DIM c(3)
30 CLS:MODE 2
40   INK 0,13:INK 1,0
50   INPUT"valores maximos de x,y";mx,my
60   INPUT"c1,c2,c3";c(1),c(2),c(3)
70   xc=320
80    x=-c(2)/(2*c(1))
90    yv=c(1)*x↑2+c(2)*x+c(3)
100   IF c(1)<0 THEN ya=0
110   IF c(1)>0 THEN ya=my
120    xl1=320:xl2=320:xr1=320:xr2=320
130    x=x-1
140    n=3
150    y=c(1)
160    FOR i=2 TO n
170      y=y*x+c(i)
175    NEXT i
180      IF c(1)<0 THEN yb=yv-y
190      IF c(1)>0 THEN yb=my-(y-yv)
200      xl2=xl2-1
210      xr2=xr2+1
220      MOVE xl1,ya
230      DRAW xl2,yb
240      MOVE. xr1,ya
250      DRAW xr2,yb
260    ya=yb
270    xl1=xl2
280    xr1=xr2
285    IF ya=1 THEN 300
290    GOTO 130
300    c(1)=c(1)+0.005:c(2)=c(2)+0.005:c(3)=c(3)+0.005
310 GOTO 80
```

```
VALORES MAXIMOS DE X E Y? 360,300
C1,C2,C3? ,004,,004,,004
```

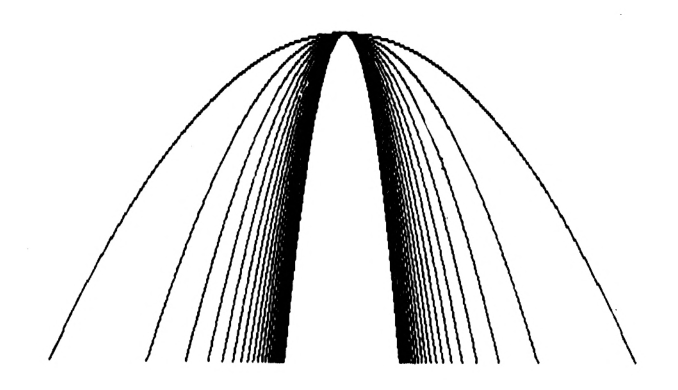


Figura 2,8 Familia de curvas parabólicas dibujadas con el programa PARABOLA

2.6 Animación de vectores

Las líneas gráficas suelen conocerse como "vectores", y la animación de líneas se denomina habitualmente "animación de vectores". Si se desea animar dibujos compuestos a base de líneas, puede resultar frustrante intentarlo con un programa en BASIC. La razón de ello es que ni un BASIC tan veloz como el del Amstrad es capaz de dibujar y borrar más que un cierto número de líneas sin que se produzca parpadeo de la imagen. El pequeño programa que ofrecemos a continuación desplaza una flecha por la pantalla. Intente alterar el tamaño de la flecha y de los pasos que ésta efectúa. Podrá comprobar que cuanto más grande sea la flecha más lentos serán sus movimientos.

Programa VECTOR

```
10 REM***programa animacion de vector***
20 REM mueve una "flecha" que atraviesa la pantalla
25 CLS
30     INK 0,13:INK 1,0
40     INPUT"tamaño de la flecha";n
50     INPUT"paso del desplazamiento";m
60 MODE 2
70     FOR x=1 TO 640 STEP m
80     GOSUB 200:REM borrado de la flecha
90 FRAME
100    GOSUB 300:REM dibujo de la flecha
```

```
110   NEXT x
120 END
200 REM subrutina para borrar la flecha
210 MOVE x,200
220 DRAW x+n,190,1,1
230 DRAW x,180,1,1
240 DRAW x,200,1,1
250 RETURN
300 REM subrutina para dibujar y borrar la flecha
310 MOVE x,200
320 DRAW x+n,190
330 DRAW x,180
340 DRAW x,200
350 RETURN
```

Podemos ver que este programa emplea la orden FRAME para sincronizar el trazado de las líneas con el refresco de la pantalla. Intente ejecutar el programa sin FRAME y comprobará que la calidad de la imagen disminuye.

Observe que este programa de dibujo de vectores emplea el modo de tinta "Xor" para borrar la imagen existente antes de dibujar un nueva imagen desplazada en pantalla.

2.7 Fractales

Concluiremos este breve capítulo con un pequeño "divertimento" matemático. La geometría fractal es una rama especializada de la geometría que no se ocupa de los entes de una, dos o tres dimensiones, sino de los situados en la "tierra de nadie" entre esas dimensiones. Seguramente habrá tenido ocasión de contemplar las hermosas huidas a través de bosques y cordilleras tridimensionales que aparecen en algunas películas, en especial en "La Guerra de las Galaxias", del equipo Lucasfilm. Esas imágenes constituyen el máximo exponente de los gráficos por ordenador, y han sido generadas con la ayuda de los más potentes ordenadores gráficos. Todas estas imágenes están confeccionadas a base de curvas fractales, y aunque la potencia de cálculo y el bagaje matemático necesarios para generarlas superan con mucho la orientación de este libro, puede utilizarse un poco de aritmética y una sencilla instrucción de dibujo de puntos para crear nuestros propios fractales de dimensiones comprendidas entre uno y dos.

El siguiente programa es una modificación de otro escrito por Greg Turk. En términos matemáticos, el tipo de fractal producido por este programa es el resultado del comportamiento de los puntos del plano descrito por la función x + iy, donde x e y son números reales e *i* es la raíz cuadrada de -1. El efecto de este programa (Figura 1.15) se produce por iteración: se resuelve la función repetidamente para cada valor de x y de y. Para

obtener los puntos x e y de la función, han de calcularse las siguientes ecuaciones:

x nuevo punto = LA * antigua X * (1 - X)
y nuevo punto = LA * antigua Y * (1 - Y)

LA es una constante de la ecuación. Puede experimentarse con LA para obtener diferentes curvas fractales. Puede ajustarse también el tamaño de la figura variando SC. Para las imágenes de la Figura 2.9, SC = 2. Cuanto menor sea SC, más rápidamente se dibujará la figura. Prepárese para esperar unos diez o quince minutos: este programa maneja bastantes cálculos.

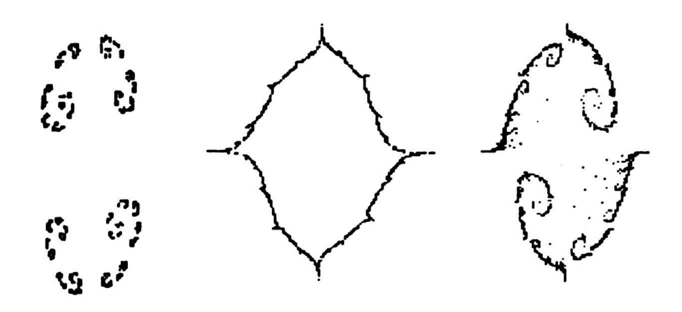


Figura 2,9 Curvas fractales de dimensiones comprendidas entre uno y dos, obtenidas con el programa FRACTAL

Programa FRACTAL

```
10 REM ****PROGRAMA FRACTAL****
20 REM muestra como puntear para generar figuras abstractas
30 RANDOMIZE
40 MODE 1
50 INK 0,0:INK 1,24
60 CX=320:CY=200
70 X=0.50001:Y=0
80 GOSUB 390
90 FOR I=1 to 10:GOSUB 310:next I
100 GOSUB 460
110 GOSUB 310
```

```
120 GOTO 100
130 END
140 REM raiz cuadrada de X,Y
150 T=Y
160 S=SQR(abs(X*X+Y*Y))
170 Y=SQR(abs((-x+s)/2))
180 X=SQR(abs((x+S)/2))
190 IF T<0 THEN X=-X
200 RETURN
210 REM cuatro por L
220 S=LX*LX+LY*LY
230 LX=4*LX/S
240 LY=-4*LY/S
250 RETURN
260 REM X,Y veces L
270 TX=X:TY=Y
280 X=TX*LX-TY*LY
290 X=TX*LX-TY*LY
300 RETURN
310 REM funcion de X,Y
320 GOSUB 260
330 X=1-X
340 GOSUB 140
350 IF RND(1)<0.5 THEN X=-X:Y=-Y
360 X=1-X
370 X=X/2:Y=Y/2
380 RETURN
390 REM entrada de valores
400 INPUT"valor de lambda";LX,LY
410 GOSUB 210
420 INPUT"valor de escala";SC
430 SC=2*CX/SC
440 CLS
450 RETURN
460 REM dibuja X,Y
470 PLOT SC*(X-0.5)+CX,CY-SC*Y
480 RETURN
```

Capítulo 3

Estructuras de datos para gráficos

3.1 Entrada de datos

Hemos visto ya que cualquier punto de la pantalla puede definirse por sus coordenadas *x,y*. Dibujar un solo punto o trazar una línea es, pues, algo extremadamente fácil. Dibujar un grupo de puntos es también muy sencillo. Pero, ¿qué sucedería si hubiéramos de dibujar una figura compleja formada por 50 o incluso 100 líneas, algunas de las cuales podrían no estar relacionadas de forma secuencial?

Necesitamos, pues, un procedimiento estándar para introducir y conservar los datos. Tres son los parámetros que deben especificarse:

(1) Las coordenadas *x,y* de todos los puntos de la figura.
(2) El orden de los puntos (es decir, la secuencia en que van a dibujarse).
(3) Las conexiones entre los puntos (¿dos puntos consecutivos van a quedar unidos o no?)

El estudio de las estructuras de datos que manejan los ordenadores es una materia de vital importancia en la ciencia informática. Aunque las técnicas gráficas avanzadas se basan generalmente en estructuras de datos de gran complejidad, en este libro nos dedicaremos a estudiar únicamente las estructuras en tablas o matrices. En muchos aspectos, esta es la más rudimentaria de las estructuras de datos, pero, al ser la única que soporta el lenguaje BASIC, nuestras posibilidades de elección no son mucho más amplias.

Tomemos en primer lugar dos matrices unidimensionales X(1 .. n), Y(1 .. n), siendo n el total de puntos a dibujar. Por supuesto, X e Y deben dimensionarse por igual, ya que todo punto tiene coordenadas x e y. Llamaremos coordenadas a los datos de estas matrices. Como X(1) precede a X(2), estas matrices nos permiten también ordenar los datos: así, el punto X(1),Y(1) se dibuja antes del X(2),Y(2), y así sucesivamente.

A continuación hemos de considerar las conexiones entre los puntos. Esta información la proporciona una tercera matriz, en este caso bidimensional. La matriz de líneas se dimensiona como V(1 .. 2,1 .. i), donde i es el número de líneas a dibujar en la figura. Ahora la primera dimensión de la

matriz W indica que para cada número de línea hay dos datos, como puede verse en la siguiente tabla. Estos dos datos no son coordenadas propiamente dichas, sino índices. En la jerga informática, un índice es simplemente un puntero que señala a otro elemento de información en el interior del ordenador. En este caso, cada índice apunta a un elemento de las matrices X e Y. El primer índice para cada línea corresponde a las coordenadas del punto de comienzo de la línea, y el segundo índice se refiere a las coordenadas finales. Así, los datos completos necesarios para dibujar un cuadrado tendrán este aspecto:

i	*XX(i)*	*Y(i)*	*W(1,i)*	*W(2,i)*
1	50	150	1	2
2	150	150	2	3
3	150	50	3	4
4	50	50	4	1

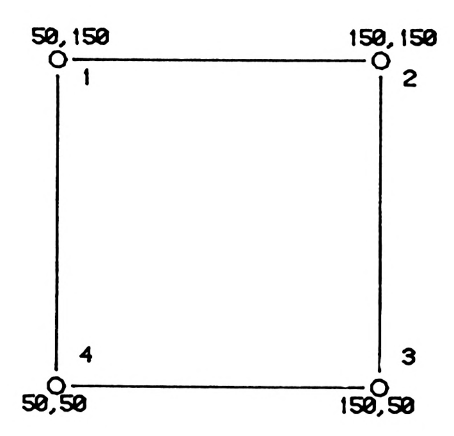


Figura 3,1 Puntos y coordenadas para un cuadrado

Observe que, en este ejemplo, X, Y y W tienen el mismo número de puntos, pero esto no tiene por qué suceder siempre. Si dibujamos la figura que aparece a continuación, la matriz W ha de contener un corte, como puede verse en los datos que aparecen debajo de la misma.

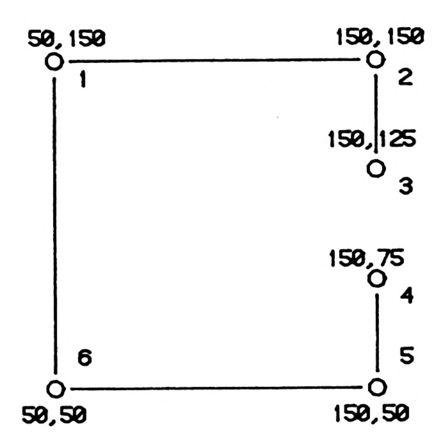


Figura 3.2 Puntos y coordenadas para un cuadrado con un 'corte'

i	*X(i)*	*Y(i)*	*V(1,i)*	*V(2,i)*
1	50	150	1	2
2	150	150	2	3
3	150	125	4	5
4	150	75	5	6
5	150	50	6	1
6	50	150		

La siguiente sección de este programa debería leer los datos de este rectángulo y dibujarlo. Observe las dos variables NPTS y LI. Especifican, respectivamente, el número de puntos y el número de líneas. A lo largo del libro emplearemos variables como estas.

Programa DIBUJOFACIL

```
10 REM **** PROGRAMA DIBUJOFACIL ****
20 REM Demostración del trazado de líneas y puntos a partir de los datos
almacenados en matrices grabadas
30 CLS
40 REM Definición de la matriz de puntos
50 READ npts
60 DIM x(npts),y(npts)
70 FOR i=1 TO npts
80 READ x(i),y(i)
90 NEXT i
```

```
100 READ li
110 DIM ln(2,li)
120 FOR i=1 TO li
130 READ ln(1,i),ln(2,i)
140 NEXT i
150 REM Ahora dibuja la figura
160 FOR i=1 TO li
170 MOVE x(ln(1,i)),y(ln(1,i))
180 DRAW x(ln(2,i)),y(ln(2,i))
190 NEXT i
200 STOP
210 DATA 6,100,100,300,100,300,150,300,250,300,300,100,300
220 DATA 5,1,2,2,3,4,5,5,6,6,1
```

Empleando las matrices X, Y y W para almacenar la información relativa a la figura, puede construirse cualquier trazado de líneas. Pueden usarse también líneas de distintos colores, siempre que no estén dentro del mismo bloque de 8 x 8 bits. Este modo de introducir la información, por medio de líneas DATA en el propio programa, dista mucho de ser óptimo, ya que es preciso cambiar una parte del programa cada vez que se quiera emplear un nuevo conjunto de datos. Es una tarea trivial para un pequeño programa de diez líneas como el anterior, pero ¿qué sucedería si se tratara de un paquete gráfico profesional, protegido tanto para lectura como para escritura? La mejor solución consiste en emplear un fichero secuencial en el que guardar los datos necesarios para dibujar la figura. Debemos, pues: (1) escribir un pequeño programa que cree el fichero de datos, y (2) habilitar nuestro programa o programas de dibujo para acceder a los datos contenidos en el fichero secuencial creado.

El programa FICHERO2D que se muestra a continuación permite introducir las matrices de datos X, Y y W en un fichero secuencial. El programa comienza solicitando el nombre que se quiere dar al fichero secuencial a crear.

Las secciones del programa para leer y escribir en un fichero secuencial son las mismas para todos los ordenadores Amstrad. Al BASIC de Amstrad le preocupa muy poco si dispone de una cinta o de una unidad de disco: la máquina operará según lo que tenga conectado. En este libro nos limitaremos a los siguientes comandos de entrada y salida:

```
OPENIN "NOMBRE DEL FICHERO
INPUT#9, VARIABLES
CLOSEIN

OPENOUT "NOMBRE DEL FICHERO
PRINT#9, VARIABLES
CLOSEOUT
```

Un toque de atención para los usuarios de unidad de disco: si ya existiera en la unidad otro fichero secuencial con el mismo nombre, se escribirá encima de él, por lo que quedará destruido. El BASIC de Amstrad proporciona, no obstante, un cierto grado de seguridad: el fichero existente queda almacenado con el nombre NOMBFICH.BAK. A pesar de ello, si se guarda una tercera versión del fichero, la versión NOMBFICH.BAK se perderá, ya que será destruida por la segunda versión .BAK de seguridad.

Programa FICHERO2D

```
10 REM****PROGRAMA FICHERO2D****
20 REM Programa para almacenar datos coordenados para dibujarlos con
DRAW2D
25 CLS
30 INPUT"NOMBRE DEL FICHERO";h$
40 OPENOUT h$
50 INPUT"NUMERO DE PUNTOS";npts
55 WRITE #9,npts
60 PRINT "INTRODUZCA LOS PARES X,Y"
70 FOR i=1 TO npts
80 INPUT"X=";x:INPUT"Y=";y
90 WRITE #9,x
100 WRITE #9,y
110 NEXT i
120 INPUT"NUMERO DE LINEAS";li
130 WRITE #9,li
140 PRINT"NUMERO DE PUNTOS DE UNION"
150 FOR i=1 TO li
160 INPUT "COMIENZO DESDE EL NUM";sn:INPUT"HASTA EL NUM";fi
170 WRITE #9,sn
180 WRITE #9,fi
190 NEXT i
200 CLOSEOUT
210 END
```

Observe que a FICHERO2D no tienen por qué preocuparle en absoluto las estructuras de datos, ya que su única misión es transferir una secuencia de números desde la unidad de disco o de cinta. El listado que aparece a continuación corresponde a un programa que dibuja una figura a partir de los datos procedentes de un fichero secuencial creado por FICHERO2D. Como puede verse, este programa (DIBUJO2D) es el "inverso" de FICHERO2D, puesto que retoma los datos importantes, rellenando con ellos las matrices X, Y y W. Estas tres matrices se dimensionan de acuerdo con los valores de las variables NPTS y LI del fichero secuencial.

Programa DIBUJO2D

```
10 REM****PROGRAMA DIBUJO2D****
20 REM genera un dibujo a partir de datos almacenados en formato
fichero2d
```

```
30 INPUT "NOMBRE DEL FICHERO";h$
40 OPENIN h$
50 REM define las matrices de puntos
60 INPUT #9,npts
70 DIM x(npts),y(npts)
80 FOR i=1 TO npts
90 INPUT #9,x(i):INPUT #9,y(i)
100 NEXT i
110 INPUT #9,li
120 REM define las matrices de lineas
130 DIM ln(2,li)
140 FOR i=1 TO li
150 INPUT #9,ln(1,i),ln(2,i)
160 NEXT i
170 CLOSEIN
180 REM ahora dibuja la figura
190 CLS
200 FOR i=1 TO li
210 MOVE x(ln(1,i)),y(ln(1,i))
220 DRAW x(ln(2,i)),y(ln(2,i))
230 NEXT i
240 END
```

El único truco de programación de DIBUJO 2D se encuentra en las líneas 210-220, que se encargan del dibujo en pantalla. Teniendo en cuenta que cada línea se dibuja entre dos pares de coordenadas X,Y, podemos deducir que las coordenadas X e Y han de venir dadas esta línea del programa. Sin embargo, lo que en realidad necesitamos saber es la serie de LINEAS que han de dibujarse. Hemos visto ya que los PUNTOS de comienzo y de término para cada línea *i* vienen dados por W(1,i) y W(2,i). Queda claro, pues, que el elemento de la matriz X(W(1,i)) es la coordenada de comienzo de la línea 1, y que X(W(2.i)) es la coordenada X del punto de término de la misma línea. El bucle FOR/NEXT entre las líneas 200 y 230 dibuja de forma secuencial todas las líneas de la figura, accediendo a las coordenadas X e Y a través de los índices de las matrices X e Y a los que apuntan los elementos de la matriz W.

3.2 Conjuntos de datos más complejos

¿Cuántas dimensiones?

En los casos más sencillos que hemos visto hasta ahora existía una equivalencia directa entre los datos de las coordenadas de nuestro fichero y las coordenadas que se dibujaban en pantalla. Con datos de dos dimensiones suele ser siempre posible establecer esta correspondencia biunívoca, aunque quizá sea necesario algún tipo de ajuste de "escala" para que los datos queden bien presentados en pantalla - en el capítulo siguiente veremos cómo hacerlo. Los datos de tres dimensiones exigen un

cuidado especial: el dato está definido por unas coordenadas X, Y y Z, mientras que la pantalla tiene sólo dos dimensiones, con lo cual hemos de "perder" de nuevo una dimensión. En el Capítulo 7 veremos cómo efectuar esta transformación. Por el momento, nos basta advertir que el dato Z se guarda en forma de una tercera matriz unidimensional asociada a las otras dos, X e Y.

Dibujo de segmentos

Lo más importante en este momento es la posible naturaleza modular de la figura que se desea dibujar. Si consideramos una pantalla repleta de información como una sola entidad constituida por informacion procedente de las matrices X, Y y Z, tendremos un conjunto estático de datos cuyo aspecto puede ser muy hermoso, pero cuya utilidad es ciertamente limitada. ¿Qué sucedería si intentásemos interaccionar de alguna forma con la figura? Quizá deseemos desplazar una parte de la figura a una posición diferente de la pantalla, o tal vez queramos borrar una parte de ella para mejorarla.

Para poder considerar las partes de la figura por separado del conjunto, debemos introducir un nuevo concepto, el segmento. Un segmento de una figura es una sección de la misma que puede tratarse de forma independiente.

Si toda la imagen va a tratarse como un solo segmento, nuestras matrices X, Y y W serán más que suficientes para almacenar y contruir el segmento. Si ha de visualizarse más de un segmento, habrá que almacenar los segmentos en series de matrices (X1,Y1,Z1 ... Xn,Yn,Zn), o bien guardarlos como bloques en unas mismas matrices X, Y, Z. Este último método es más elegante y se emplea con más frecuencia, por lo que es el apropiado para nosotros. Pero, ¿de qué forma definimos los segmentos si todos ellos se encuentran en la misma matriz? Veámoslo con un ejemplo gráfico.

En el diagrama se muestra una sencilla escena constituida por cuatro segmentos diferentes, una mesa, cuatro sillas, un televisor y una lámpara. Supondremos que esta estructura es parte de un programa de diseño de decoración, en el que el usuario puede mover los elementos por la pantalla como desee.

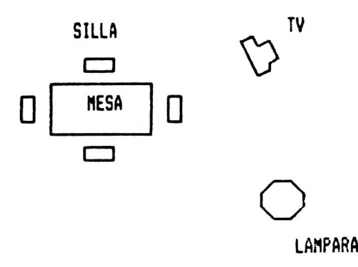


Figura 3.3 Figura construida a base de segmentos

Las matrices de almacenamiento de los datos que configuran el segmento pueden tener un aspecto como este:

i	*X(i)*	*Y(i)*	*W(1,i)*	*W(2,i)*
6 - 10		MESA	DATOS	
11 - 19		LAMPARA	DATOS	
20 - 28		*TV*	*DATOS*	

Observe que, aunque hay un sólo grupo de datos para la silla, hay cuatro sillas en nuestra escena. La capacidad para recuperar los datos de un segmento en el que han sido copiados los segmentos individuales nos muestra otro aspecto de su potencia. Para poder acceder a los datos que representan un segmento determinado, es preciso conocer dónde comienza la primera línea de cada segmento, por ejemplo, la silla, y dónde termina su última línea. Para ello se define una nueva matriz bidimensional, que se dimensiona como S(2,NS), donde NS representa el número total de segmentos, cuatro en este caso. Este será el aspecto de la matriz S para nuestro ejemplo de la habitación:

i	*S(1,i)*	*S(2,i)*
1	1	6
2	7	11
3	12	20
4	21	29

Así, S(1,i) es el índice para la línea de comienzo del segmento i-ésimo, y S(2,i) es el de la línea final de este mismo segmento. Vemos ya que nuestras estructuras de datos se están empezando a embrollar. Tenemos un sistema de punteros en tres niveles, en el que S apunta a W, la cual a su vez apunta a las matrices X e Y.

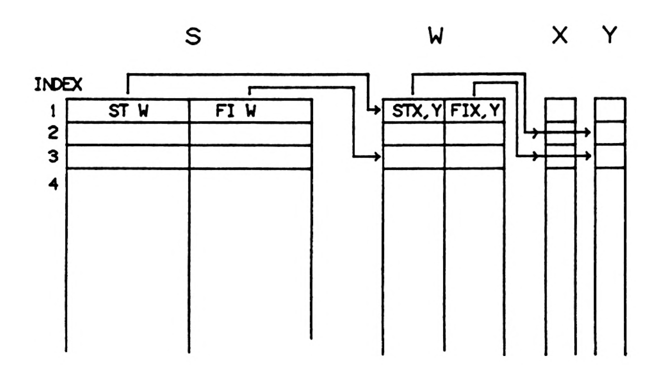


Figura 3.4 Relación entre las matrices S,W,X e Y. Observe que las matrices S y W contienen índices hacia las posiciones de las matrices a las que apuntan. X,Y COM = cordenadas de comienzo de la línea. X,Y FIN = coordenadas finales de la línea.

Ya tenemos toda la información necesaria para acceder al segmento desde el programa principal que genera la figura.

3.3 Manipulación de segmentos

No apreciaremos toda la potencia de los segmentos hasta que aprendamos a manipular (es decir, cambiar de tamaño y desplazar) los elementos de la imagen; la descripción de estas manipulación constituye la base del Capítulo 4. Así, por ejemplo, para visualizar segmentos en lugares variables de la pantalla se emplea la técnica de traslación. Es posible, no obstante, con los conocimientos actuales, construir nuestra propia escena "segmentada"; pueden añadirse las siguientes secciones de código a FICHERO2D y DIBUJO2D, respectivamente. Estas mejoras permiten, en efecto, que los programas hagan algo más de lo que serían capaces sin ellas, pero además nos muestran cómo la matriz S se define y emplea de forma elemental.

```
200 REM Ampliaciones de FICHERO2D
210 INPUT "Segmentos?";sn
220 PRINT #9,sn
230 PRINT"Lectura de la matriz de datos S"
240 FOR i=1 TO sn
250 INPUT j,k,l
260 PRINT #9,j:PRINT #9,k:PRINT #9,l
270 NEXT i
280 CLOSEOUT:END
```

```
240 REM Ampliaciones de DIBUJO2D
250 INPUT #9,sn
260 DIM s(3,sn)
270 FOR i=1 TO sn
280 INPUT #9,s(1,i),s(2,i),s(3,i)
290 NEXT i
300 REM Ahora dibuja el segmento; cambie el valor de seg para dibujar
otro segmento distinto del 1
320 FOR i=s(1,seg) TO s(2,seg)
330 MOVE x(w(1,i)),y(w(1,i))
340 DRAW x(w(2,i)),y(w(2,i))
350 NEXT i
360 END
```

NOTA - Todas las secciones que se puedan añadir a programas ya existentes podrían provocar la destrucción de algunas líneas de éstos si las numeraciones de sus líneas coinciden. Salvo especificación en otro sentido, todas las líneas de numeraciones no solapadas del programa original y del insertado han de incluirse en la nueva versión del programa. La forma más fácil de hacerlo es: (1) crear un fichero de disco

o de cinta que contenga las adiciones; (2) cargar el programa original en memoria; y (3) mezclar (MERGE) las adiciones con el programa principal.

3.4 La forma más fácil de dibujar

Hemos visto hasta ahora cómo producir un conjunto de datos con los que representar segmentos únicos o múltiples. El problema es, por supuesto, que los datos de los puntos deben calcularse laboriosamente a mano. Debe existir un método mejor. En esta sección veremos un programa que permite componer los ficheros de datos de dos dimensiones en los ratos libres, mediante la pantalla y un *joystick*. Este programa se llama SKETCH, y es posterior al original, llamado Sketchpad: una de las primeras herramientas de diseño asistido por ordenador, creado por Ivan Sutherland a comienzos de los años 60.

SKETCH utiliza un *joystick* para desplazar el cursor por la pantalla. El flujo del programa lo controlan las siguientes acciones:

Tecla pulsada	Efecto
botón de disparo	Comienzo de la línea
botón de disparo	Fin de la línea
B	Siguiente punto aislado
E	Fin de segmento
F	Fin de la figura
S	Comenzar nuevo segmento

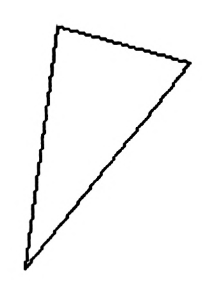
Esta información de control se muestra en pantalla en la versión de SKETCH que se ofrece a continuación. Una vez familiarizado con los comandos, quizá crea conveniente suprimir la sentencia GOSUB (línea 135) de llamada a la rutina de texto, para que no se impriman en pantalla las instrucciones. El volcado de pantalla de la página siguiente nos ilustra este aspecto.

Una vez marcado el inicio de una línea, el cursor va moviendo la línea alrededor del punto de comienzo. Ello nos permite ver el efecto que produce la colocación de una línea en cualquier posición. Recuerde de nuestra discusión acerca de la técnica de inversión de pixels, al final del capítulo anterior, que gracias a esta técnica es posible mover partes de la imagen por encima de otros elementos situados debajo sin borrar estos últimos. Por eso, todas las órdenes LINE del programa SKETCJ toman la forma de inversión, en lugar de la de escritura/borrado.

A continuación ofrecemos el programa SKETCH. Es considerablemente más largo que ninguno de los programas-ejemplo vistos hasta ahora, pero podrá comprobar que su uso ahorra mucho más tiempo en la introducción de los datos del que supone teclearlo completo.

PROGRAMA SKETCH

OTRO SEGMENTO?



DISPARO=PRINC/FIN DE LINEA   B=CORTAR LINEA
F=FIN   S=SIG SEG   E=ACABAR SEGMENTO

Figura 3.5 Volcado de pantalla de SKETCH. Se ha dibujado un triángulo con el *joystick*

Programa SKETCH

```
5 REM****PROGRAMA SKETCH****
10 REM definicion del tamaño y los pasos del cursor
20 cs=2:ss=2
30 MODE 1
40 INPUT"NOMBRE DEL FICHERO DE DATOS";n$
45 CLS
50 REM ajuste de contadores y banderas
60 fl=0:npts=1:na=1
70 lb=0:REM contador de lineas
80 se=0:REM bandera indicadora de fin de segmento
90 sl=0:REM contador de segmentos
92 fi=0:REM bandera indicadora de fin de linea
93 li=0:REM contador de linea del segmento
94 jy=1:REM bandera de comienzo/fin de linea
100 DIM xp(500),yp(500),ln(2,500),s(3,10):REM dimensionamiento de matrices
120 REM colocacion del cursor en la posicion central
130 x=320:y=200
135      GOSUB 3500:REM colocacion de texto en pantalla
140      GOSUB 180:REM rutina de dibujo del cursor
150      GOSUB 230:REM rutina de movimiento del cursor
160      GOSUB 340:REM rutina de barrido de la linea
170 GOTO 140
```

```
180 REM rutina de dibujo del cursor
190 x1=x-cs:y1=y-cs:x2=x+cs:y2=y+cs
200      MOVE x1,y
205      DRAW x2,y,1,1
210      MOVE x,y1
215      DRAW x,y2,1,1
220 RETURN
230 REM rutina de movimiento del cursor
240 y3=y:x3=x
250      IF JOY(0)=0 THEN 310
260      IF JOY(0)=1 THEN y=y+ss:GOTO 310
270      IF JOY(0)=2 THEN y=y-ss:GOTO 310
280      IF JOY(0)=4 THEN x=x-ss:GOTO 310
290      IF JOY(0)=8 THEN x=x+ss:GOTO 310
310          MOVE x3,y2
320          DRAW x3,y1,1,1
325          MOVE x1,y3
326          DRAW x2,y3,1,1
330 RETURN
340 REM rutina de dibujo y barrido de la linea
350 a$=INKEY$
355 IF a$="" AND JOY(0)<>16 THEN IF fl=0 THEN RETURN
370 IF JOY(0)=16 AND jy=1 THEN jy=2:LOCATE 1,1:PRINT "S":GOSUB 3000:GOTO
430
380 IF JOY(0)=16 AND jy=2 THEN jy=1:LOCATE 1,1:PRINT "F":GOSUB 3000:GOTO
460
390 IF a$="B" OR a$="b" THEN jy=1:GOTO 450:REM corte de la linea
400 IF a$="E" OR a$="e" THEN se=1:jy=1:GOTO 460:REM fin de la figura
410 IF fl=0 THEN RETURN
420 GOTO 650:REM dibujo/borrado de la linea
430 xi=x:yi=y:REM coordenadas de comienzo
440 fl=1:RETURN
450 fi=1:REM bandera de corte de linea
460 xf=x:yf=y:REM entrada de un punto
480    MOVE xi,yi
485    DRAW xf,yf
490 npts=npts+1:na=na+1:li=li+1:lb=lb+1:REM incrementar contadores
500 xp(na)=xf:yp(na)=yf:REM entrada de puntos
510 xp(na-1)=xi:yp(na-1)=yi:REM entrada de puntos
560 ln(1,lb)=na-1:REM entrada de indices de lineas
570 ln(2,lb)=na
580 IF fi=1 THEN na=na+1:fi=0:REM incremento si la bandera de corte de
linea activada
590 IF se=1 THEN s1=s1+1:s(1,s1)=npts-li:s(2,s1)=npts-1:s(3,s1)=0:GOTO 690
630 fl=0:RETURN
640 fl=0
650 REM Dibujo/borrado de la linea
660    MOVE x,y
665    DRAW xi,yi,1,1
670    MOVE x,y
```

```
675    DRAW xi,yi,1,1
680 RETURN
690 REM continuar
710 FOR i=s(1,s1) TO s(2,s1)
730    MOVE xp(ln(1,i)),yp(ln(1,i))
735    DRAW xp(ln(2,i)),yp(ln(2,i)),1,0
740 NEXT i
750 k$=INKEY$:LOCATE 1,2:PRINT"otro segmento?"
755 IF k$="" THEN 750 :REM hasta que no se pulse una tecla, vuelve a 750
757 LOCATE 1,2:PRINT"                    "
770 IF k$="F" THEN 800:REM no, entonces termina
780 li=0:fl=0:se=0:na=na+1:REM si, entonces actualiza los contadores
790 RETURN
800 REM ahora crea el fichero que contiene los datos
810 OPENOUT n$
840 WRITE #9,na
850   FOR i=1 TO na
860     WRITE #9,xp(i)
870     WRITE #9,yp(i)
875   NEXT i
880 WRITE #9,lb
890   FOR i=1 TO lb
900     WRITE #9,ln(1,i)
910     WRITE #9,ln(2,i)
915   NEXT i
920 WRITE #9,s1
930   FOR i=1 TO s1
940     WRITE #9,s(1,i)
950     WRITE #9,s(2,i)
960     WRITE #9,s(3,i)
965   NEXT i
970 CLOSEOUT
980 END
3000 FOR i=1 TO 200:NEXT i:RETURN
3500 LOCATE 13,1:PRINT "PROGRAMA SKETCH";
3510 LOCATE 1,24:PRINT" DISPARO=PRINC/FIN DE LINEA  B=CORTE DE LINEA"
3520 LOCATE 1,25:PRINT"     F=FIN  S=SIG SEGMENTO   E=FIN DE SEGMENTO"
3530 MOVE 0,50
3540 DRAW 640,50
3600 RETURN
```

SKETCH puede descomponerse como sigue:

LíNEAS		
	5 - 130	PREPARACION DE LA PANTALLA, EL CURSOR Y LAS MATRICES
	135	ESCRITURA DE LAS INSTRUCCIONES EN LA PANTALLA
	140 - 170	BUCLE PRINCIPAL
	180 - 330	LECTURA DE LAS TECLAS DE CONTROL
	340 - 400	COMIENZO DE LINEA

450	INDICADOR PARA EL CORTE DE LA LíNEA
480	DIBUJO DE LA LINEA
490	ACTUALIZACION DE CONTADORES
500 - 570	COLOCACION DE LAS COORDENADAS EN LAS MATRICES X,Y,W
580	REINICIALIZACION DE LOS CONTADORES PARA EL CORTE DE LA LINEA
590	AJUSTE DEL CONTADOR DE SEGMENTOS
650 - 680	BORRADO/ESCRITURA TEMPORAL DE LA LINEA
710 - 740	REPETICION DE UN SEGMENTO
750 - 790	¿OTRO SEGMENTO? EN CASO AFIRMATIVO, REINICIALIZA LOS CONTADORES
800 - 980	CREACION DE UN FICHERO DE DATOS
3000	SIMPLE BUCLE DE RETARDO
3500 - 3600	RUTINA DE ESCRITURA DE TEXTO

La sección de control del cursor de SKETCH utiliza el *joystick* porque este dispositivo proporciona un control de la generación de la imagen mucho mejor que el que permite el control por teclado.

El uso de las matrices X,Y,W y S en SKETCH es básicamente el mismo que para los programas que ya hemos visto. Lo verdaderamente nuevo de SKETCH es la manipulación del *joystick* y el cursor para la creación interactiva de datos. Las líneas 180 - 330 de SKETCH gobiernan el movimiento del cursor, en el que intervienen tres pasos, principalmente, a saber:

(1) Dibujo del cursor
(2) Movimiento del cursor
(3) Borrado del cursor

El cursor se dibuja mediante dos líneas rectas, una horizontal y la otra vertical, que se cortan en sus puntos medios. La longitud de estas líneas se ajusta mediante la variable CS. Si el punto a dibujar se encuentra en las coordenadas x,y, la línea X se dibujará desde *x-CS,y* hasta *x+CS,y*; la línea Y se dibujará, por tanto, desde *x,y-CS* hasta *x,y+CS*. El movimiento del cursor tiene lugar empleando los valores que entrega el joystick, para incrementar o decrementar los valores de x o y, con lo cual puede colocarse el cursor en una nueva posición alrededor del punto x,y original. Para borrar el cursor, éste vuelve al punto anterior alrededor del punto x,y original, pero en este caso cada pixel es borrado, en lugar de pintado. Combinando estas dos operaciones, el movimiento del cursor por la pantalla puede controlarse suavemente con el *joystick*.

3.5 Cómo utilizar el programa SKETCH

Al principio, SKETCH puede parecer difícil de utilizar. Dibujar "por control remoto" no es algo fácil de aprender, pero pueden obtenerse resultados bastante impresionantes, como se ve en las figuras de la página siguiente. Una vez adquirida la habilidad necesaria para pulsar la

tecla correcta o el botón de disparo en el momento apropiado, este programa puede servirnos para producir datos muy útiles. Los dibujos que aquí se muestran fueron obtenidos empleando un truco adicional. Cada silueta se dibujó previamente en una transparencia de formato DIN A-4 (que puede comprarse en cualquier tienda de arte o de material de oficina). A continuación se pegó la película sobre la pantalla del monitor mediante cinta adhesiva. De esta forma fué muy fácil trazar el dibujo con un *joystick* y el programa SKETCH. Intente crear un mapa de algunos estados norteamericanos, considerando a cada uno de ellos como un segmento. Pruebe a escribir una nueva versión de DIBUJO2D que permita acceder a los segmentos en cualquier orden. Puede utilizar este programa para comprobar los conocimientos geográficos de sus amigos.

MAPA DE INGLATERRA Y GALES CRÈADO MEDIANTE EL PROGRAMA SKETCH EL 5/7/85



Figura 3.6 Una figura creada con SKETCH: un mapa mudo de Inglaterra y Gales

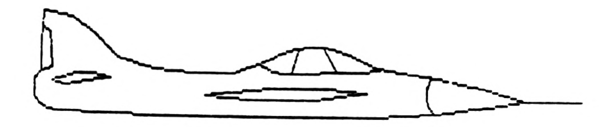


Figura 3.7 Figura creada con SKETCH: un avión a reacción

El empleo de SKETCH junto con una transparencia de esta forma es un ejemplo de digitalizador casero, es decir, un dispositivo capaz de transferir directamente los datos espaciales al interior del ordenador, Los digitalizadores comerciales pueden costar más de cuatro millones de pesetas. Ahora tenemos uno por el precio de un *joystick*.

Si tiene Vd. un CPC 6128 o un CPC 664, quizá desee añadir al programa SKETCH la siguiente rutina de rellenado. Su funcionamiento es el siguiente: una vez hecho el dibujo, debe situarse el cursor en el centro del área a colorear, pulsando a continuación la tecla Z. Puede entonces escogerse el color, pulsando las teclas 1, 2 ó 3. Este recinto (¡asegúrese de que es cerrado!) quedará coloreado, volviendo al programa principal para dibujar nuevas líneas o se seleccionar otros colores si se desea. La rutina de rellenado de superficies no afecta a la capacidad de almacenamiento de la figura, puesto que el proceso de coloreo es transparente a las estructuras que guardan los puntos y las líneas.

```
217      k$=INKEY$:IF k$="Z" THEN GOSUB 4000:REM rutina de coloreado
4000 REM rutina de coloreado para utilizar con el programa SKETCH (solo
en el CPC 464)
4010 REM utilizar solo en el modo 1
4020 INK 2,3:INK 3,12
4025 MOVE x+2,y+2
4030     k$=INKEY$
4040     IF k$="" THEN 4030
4050     IF k$="1" THEN FILL 1
4060     IF k$="2" THEN FILL 2
4070     IF k$="3" THEN FILL 3
4080 RETURN
```

PROGRAMA SKETCH

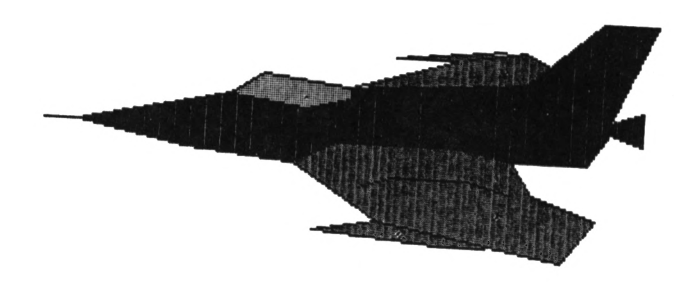


Figura 3.8 Figura obtenida con SKETCH mejorado para colorear

Capítulo 4

Manipulación de datos de 2 dimensiones

4.1 El sistema de coordenadas

En este capítulo aprenderemos a desplazar figuras de dos dimensiones y segmentos a través de la pantalla. Antes de ello, revisaremos algunas de las reglas elementales de geometría en coordenadas de dos dimensiones. Aunque ya hemos visto unos cuantos programas que emplean las coordenadas X e Y, será conveniente examinar con mayor detalle el tema de los sistemas de coordenadas.

En esencia, estamos manejando un sistema de coordenadas rectangulares, que consta de dos escalas llamadas ejes. Una de ellas es horizontal, y la otra vertical. El punto de intersección de ambas se llama origen. Es habitual asignar a la parte derecha del eje horizontal los valores positivos de la X, y a la parte superior del eje vertical los valores positivos de la Y (y, por consiguiente, las partes izquierda e inferior representan, respectivamente, las X y las Y negativas).

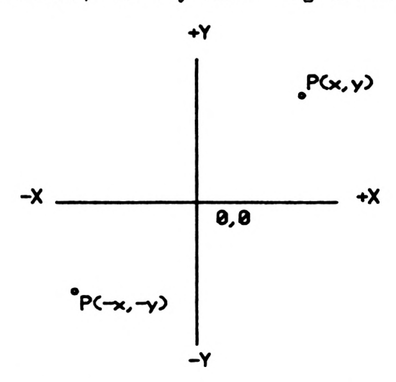


Figura 4.1 Etiquetado de los ejes en un sistema de coordenadas X,Y. Se muestran dos puntos, uno positivo y el otro negativo (con las mismas coordenadas pero de distinto signo).

El diagrama anterior muestra el trazado de los ejes. Utilizándolos podemos designar cualquier punto como P(x,y), en cualquiera de los cuatro sectores o cuadrantes alrededor del origen, según cuál sea el signo de x e y. En la mayoría de los micros domésticos (y en todos los demás ordenadores con los que este autor ha trabajado) el origen se encuentra en la esquina inferior izquierda de la pantalla. Esto implica que las coordenadas de los puntos que aparecen en pantalla han de ser positivas.

Ahora que ya hemos visto cómo representar puntos en un sistema de coordenadas rectangulares, podemos dibujar algunas figuras sencillas mediante los comandos MOVE y DRAW, junto con los datos almacenados en una estructura simple. Hasta aquí todo va bien, siempre que todos los datos queden dentro de los límites de las coordenadas de la pantalla del Amstrad (X = 0 -639, Y = 0 -199), y que no haya que efectuar ninguna manipulación espacial con los datos. Pero, ¿qué sucede si queremos mover un objeto por la pantalla, modificar su tamaño en uno o en ambos ejes, o hacerlo girar en torno a un eje especificado?

Las técnicas mediante las cuales se llevan a cabo estas operaciones se conocen como *transformaciones*, en la jerga gráfica, y las más importantes son la "santísima trinidad" de las transformaciones: *rotación*, *cambio de escala* y *traslación*. Mediante combinaciones de estas transformaciones puede realizarse cualquier manipulación con figuras bidimensionales. No obstante, si bien es posible realizar cualquier transformación de dos dimensiones con una mezcla de "fuerza bruta" aritmética y trigonometría elemental, nosotros utilizaremos el *álgebra de matrices* para todas nuestras transformaciones. El álgebra de matrices nos proporciona una herramienta extremadamente eficaz para manipular datos coordenados. Su inconveniente es que para comprender esta forma de manipulación es preciso aprender qué son las matrices, y cómo manejarlas.

No queremos decir con esto que haya nada especialmente complicado en la manipulación de las matrices; en el Apéndice 2 se ofrecen algunos detalles en este sentido. A lo largo de esta sección haremos referencia a este apéndice, pero si las matemáticas le producen verdaderas pesadillas, no se preocupe. No es necesario entender los conceptos del Apéndice 2 para emplear las rutinas que aquí se ofrecen. En efecto, es perfectamente posible trabajar a lo largo de todo el libro sin tener idea de qué son las matrices: pero no sea vago; intente acudir al Apéndice 2; como ocurre a veces con un nuevo plato de cocina extranjera, quizá acabe deleitando su paladar.

4.2 Rotación

Como su nombre sugiere, la rotación implica un giro de la figura, o de parte de ella, de un cierto ángulo en el espacio. Antes de efectuar una rotación necesitamos conocer un dato muy importante - el punto alrededor del cual va a girarse el objeto. Aunque es matemáticamente posible calcular una rotación alrededor de cualquier punto del espacio de dos dimensiones, el punto alrededor del cual es más sencillo el giro es el

origen. Utilizando el álgebra de matrices es fácil establecer una matriz A que, una vez calculada, proporcione puntos x,y girados en torno al origen a partir de un conjunto de puntos dados. Sigamos paso a paso un sencillo programa de rotación bidimensional para ver cómo trabaja.

El primer paso consiste en colocar los datos en las matrices de BASIC X,Y y W en las que han de almacenarse. Para simplificar el programa no consideraremos, de momento, la matriz de segmentos. En todas las transformaciones de dos dimensiones nuestras matrices aparecerán como tablas de DIMensión 3 x 3. Para almacenar la matriz emplearemos la tabla A(3,3), y una rutina de rotación generará los valores de la misma. Por último, multiplicaremos los datos x,y y la matriz de rotación para transformar los datos. El diagrama de flujo de nuestro programa tendrá un aspecto similar a este:

Entrada de datos x,y

↓

Entrada del ángulo de rotación

↓

Generación de la matriz de rotación

↓

Multiplicación de cada par x,y
por la matriz de rotación

↓

Dibujo de la figura utilizando
los puntos transformados

↓

Bucle para la multiplicación
de matrices

A continuación aparece el programa completo. Si lo ejecuta, podrá comprobar que dibuja una flecha que comienza en el eje X y va acercándose gradualmente al eje Y.

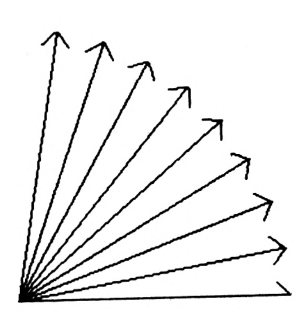


Figura 4,2 Resultado de GIRO, Cada flecha se ha girado + 10 grados en torno al origen.

Puede verse claramente que el origen está actuando como centro de giro. El programa calcula una secuencia de rotaciones de 10 grados, en un bucle que comprende las líneas 160 a 190. Observe que la matriz de transformación sólo debe generarse una vez: como todas y cada una de las rotaciones son de 10 grados, es únicamente la multiplicación de las matrices la que debe repetirse. Esta multiplicación está codificada entre las líneas 430 y 490. Observe el empleo de la constante .17455 en la línea 160. Son diez grados en radianes, puesto que un grado equivale, aproximadamente, a .017455 radianes (aproximadamente - hay 2π radianes en 360 grados).

Programa GIRO

```
10 REM****PROGRAMA GIRO****
20 REM **** programa GIRO ****
25 REM muestra la rotacion en dos dimensiones utilizando matrices de
transformacion
30 REM gira alrededor del origen 0,0
40 DIM x(4),y(4),xp(4),yp(4),ln(2,3),a(3,3),po(3),p(3)
45 CLS
50 REM captura de los datos para dibujar la figura
60     READ npts
70     FOR i=1 TO npts
80     READ x(i),y(i)
90     NEXT i
100    READ li
110    FOR i=1 TO li
120    READ ln((1,i),ln(2,i)
130    NEXT i
140 DATA 4,0,0,0,195,10,185,-10,185,3,1,2,2,3,2,4
150 REM ajuste del angulo de rotacion
160 an=an+0.174533
170    GOSUB 360:REM rutina de rotacion
180    GOSUB 430:REM dibujo de la figura
190 GOTO 160
200 END
220        a(1,1)=cos(an):a(1,2)=sin(an):a(1,3)=0
360 REM rutina de rotacion
370        a(1,1)=cos(an):a(1,2)=sin(an):a(1,3)=0
380        a(2,1)=-sin(an):a(2,2)=cos(an):a(2,3)=0
390        a(3,1)=0:a(3,2)=0:a(3,3)=1
400 RETURN
430 REM realizacion de la figura
440        FOR i=1 TO npts:p(1)=0:p(2)=0:p(3)=0
445        po(1)=x(i):po(2)=y(i):po(3)=1
450        FOR j=1 TO 3
455        FOR k=1 TO 3
460          p(j)=(a(j,k)*po(k))+p(j)
470        NEXT k:NEXT j
480        xp(i)=p(1):yp(i)=p(2)
```

```
490       NEXT i
500 REM ahora dibuja la figura
510       FOR i=1 TO li
520       MOVE xp(ln(1,i)),yp(ln(1,i))
530       DRAW xp(ln(2,i)),yp(ln(2,i))
540       NEXT i
550 RETURN
```

En GIRO, puede verse que los datos X Y originales se conservan inalterados, y que las transformaciones se llevan a cabo sobre otras dos nuevas matrices llamadas XP e YP. Es bastante habitual, en efecto, guardar los datos de los puntos a dibujar en pantalla en matrices de trabajo, que se actualizan constantemente durante la ejecución del programa. En los programas posteriores acabaremos considerando a las matrices XP e YP como algo rutinario.

Aunque hemos empleado el álgebra de matrices para determinar las coordenadas giradas, es posible utilizar, si el ángulo de rotación no es demasiado grande, un método más elemental y mucho más rápido, pero que sólo funciona para incrementos de uno o dos grados. Las nuevas coordenadas para cada punto quedan, con este procedimiento:

$$x' = x - y \text{ sen } \theta$$
$$y' = x' \text{ sen } \theta + y$$

Puede volverse a escribir el programa de rotación de la flecha para dibujar, pongamos, rotaciones de un solo grado entre 0 y 90, para comprobar el incremento de velocidad que se produce. Pero cuidado:si θ toma un valor similar a 10 grados, ¡se producirá un lío espantoso!

4.3 Traslación

Una rotación es, sencillamente, una reducción o incremento de los valores de las coordenadas X y/o Y de un segmento o de la figura completa. Las traslaciones pueden llevarse a cabo también por medio de matrices, utilizando de nuevo una tabla de 3 x 3 para almacenar la matriz. Puede mejorarse fácilmente el programa GIRO para que sea capaz de efectuar también traslaciones. La rutina de traslación se efectúa mediante la siguiente rutina, que puede añadirse a GIRO. En las próximas páginas construiremos un programa general de transformaciones de dos dimensiones, al que llamaremos, de ahora en adelante, TRV. A la primera versión podemos llamarla TRV1

```
600 rem rutina de traslación
610 a(1,1)=1:a(1,2)=0:a(1,3)=tx
620 a(2,1)=0:a(2,2)=1:a(2,3)=ty
630 a(3,1)=0:a(3,2)=0:a(3,3)=1
640 return
```

TX y TY son los cambios que deben realizarse en las coordenadas x e y de los puntos a trasladar. Estos valores deben prepararse en el programa antes de llamar a la rutina de traslación. Si el programa TRV1 debe desplazar la flecha a lo largo del eje X, en incrementos de 10 unidades, por ejemplo, habrán de efectuarse los siguientes cambios:

```
135  tx=10:ty=0:rem asignación de los valores del desplazamiento
170  gosub 600: rem paso del desplazamiento
```

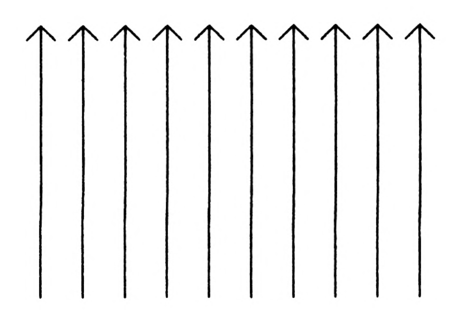


Figura 4.3 Resultado de TRV1. Cada flecha se ha desplazado +10 pixels en el eje X

En la programación gráfica en general, la traslación es un procedimiento de vital importancia. Permite desplazar imágenes o segmentos desde y hacia el origen. Veíamos en la sección anterior que las rotaciones más fáciles de realizar son las que tienen como centro de giro el origen, y mediante las traslaciones apropiadas puede simularse la rotación en torno a cualquier punto en el espacio coordenado. Para ello se emplea la siguiente secuencia:

(1) Trasladar el objeto al origen
(2) Girar el objeto
(3) Devolver el objeto a su posición original

Con leves mejoras, puede utilizarse el programa TRV1 para efectuar esta secuencia (no cabe duda de que estamos convirtiendo a TRV1 en un programa para efectuar transformaciones bidimensionales). Teniendo en cuenta que las rutinas de traslación y rotación ya están en TRV1, poco queda que añadir. Antes de echar un vistazo al empleo conjunto de la

traslación y la rotación, debemos mencionar el tercer tipo de transformación.

4.4 Cambio de escala

Quizá tenga Vd. un objeto en el centro de la pantalla y desee dilatarlo para que ocupe toda su superficie. O tal vez desee expandir un objeto horizontal o verticalmente. Estas operaciones se realizan incrementando las distancias relativas entre los puntos coordenados del segmento.

La siguiente rutina genera una matriz que puede utilizarse en este tipo de transformaciones. Su configuración es idéntica a la de las matrices de rotación y traslación: una matriz de 3 x 3 A(3,3).

```
700  rem rutina de cambio de escala
710  a(1,1)=sx:a(1,2)=0:a(1,3)=0
720  a(2,1)=0:a(2,2)=sy:a(2,3)=0
730  a(3,1)=0:a(3,2)=0:a(3,3)=1
740  return
```

Las variables sx y sy contienen los valores por los que han de multiplicarse las coordenadas x e y, repectivamente. Si uno de los ejes no sufre ningún cambio, el valor de la variable permanece a 1, no a 0. De la misma manera pueden utilizarse también reducciones.

La siguiente expansión de TRV1 (llamada TRV2) nos muestra el empleo, en el mismo programa, de todas las transformaciones examinadas hasta el momento. El programa solicita los valores de escala y de desplazamiento, y dibuja una pequeña "nave espacial", cuyas mínimas coordenadas se encuentran en 10,10 (esquina inferior izquierda). La secuencia de transformaciones del programa está comprendida entre las líneas 240-290. Ejecute este programa para familiarizarse con el efecto que pueden tener en la figura los diferentes valores de tamaño y posición de la nave espacial.

Programa TRV2

```
10 REM ****PROGRAMA TRANSFORMACION V2****
20 REM programa que efectua una traslacion, cambio de escala y giro de
una nave espacial
30 DIM x(20),y(20),ln(2,30),a(4,4),p(3),po(3)
40 REM captura de los datos para la figura
50   READ npts
60   FOR i=1 TO npts:z=3
70   READ x(i),y(i)
80   NEXT i
90   READ li
100  FOR i=1 TO li
110  READ ln(1,i),ln(2,i)
120  NEXT i
```

```
130 REM datos de la nave espacial
140 DATA 11,10,10,10,30,20,40,20,60,25,70,30,60,30,40,40,30,40,10,30,20,20,20
150 DATA 11,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,1
160 REM entrada de la informacion relativa a la transformacion
170 CLS
180   INPUT "traslacion, TX,TY";tx,ty
190   INPUT "cuantia del cambio de escala EX,EY";sx,sy
200   INPUT "rotacion en grados";an
210 REM cambio a radianes del angulo de giro
220   an=an*0.017455
230 REM ahora realiza las transformaciones
240   t1=-x(1):t2=-y(1):REM traslado al origen
250 GOSUB 360:REM rutina de traslacion
260 GOSUB 510:REM dibujo de la figura
270 GOSUB 410:REM rutina de cambio de escala
280 GOSUB 510:REM dibujo de la figura
290 GOSUB 460:REM rutina de rotacion
300 GOSUB 510:REM dibujo de la figura
310 t1=tx:t2=ty
320 GOSUB 360:REM rutnia de traslacion
330 GOSUB 510:REM dibujo de la figura
340 END
360 REM rutina de traslacion
365   a(1,1)=1:a(1,2)=0:a(1,3)=t1
370   a(2,1)=0:a(2,2)=1:a(2,3)=t2
375   a(3,1)=0:a(3,2)=0:a(3,3)=1
380 RETURN
410 REM rutina de cambio de escala
420   a(1,1)=sx:a(1,2)=0:a(1,3)=0
425   a(2,1)=0:a(2,2)=sy:a(2,3)=0
430   a(3,1)=0:a(3,2)=0:a(3,3)=1
440 RETURN
460 REM rutina de rotacion
465   a(1,1)=COS(an):a(1,2)=SIN(an):a(1,3)=0
470   a(2,1)=-SIN(an):a(2,2)=COS(an):a(2,3)=0
475   a(3,1)=0:a(3,2)=0:a(3,3)=1
480 RETURN
510 REM procesamiento de los puntos de la figura
520   FOR i=1 TO npts
525     p(1)=0:p(2)=0:p(3)=0
530     po(1)=x(i):po(2)=y(i):po(3)=1
540 FOR j=1 TO 3
545 FOR k=1 TO 3
550   p(j)=(a(j,k)*po(k))+p(j)
555 NEXT k
560 NEXT j
565   x(i)=p(1)
570   y(i)=p(2)
575 NEXT i
580 REM ahora dibuja la figura
```

```
600   FOR i=1 TO li
610     MOVE x(ln(1,i)),y(ln(1,i))
620     DRAW x(ln(2,i)),y(ln(2,i))
630   NEXT i
640 FOR i=1 TO 1000:NEXT i
650 RETURN
```

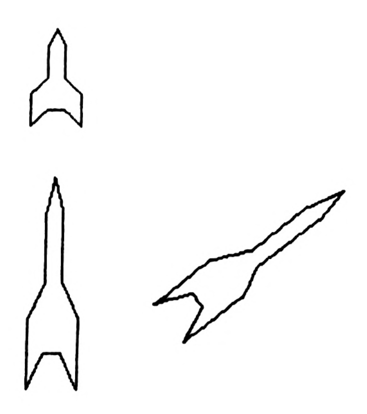


Figura 4.4 Resultado de TRV2. La nave espacial (1) se ha ampliado, (2) girado y (3) desplazado a una nueva posición.

4.5 Secuencias de transformaciones

Ya hemos cubierto las secuencias de transformaciones más sencillas: traslación al origen, rotación y retorno a la posición original. Para efectuar la secuencia de transformaciones de TRV2 empleábamos un método bastante laborioso, como mera demostración. En realidad, las etapas intermedias de traslación al origen y retorno a la posición inicial pueden llevarse a cabo sin necesidad de calcular posiciones coordenadas intermedias. Como podrá comprobar si lee el Apéndice 2, el secreto está en multiplicar entre sí, una vez creadas, las series de matrices de 3 x 3

para la rotación, el cambio de escala y la traslación. Podemos utilizar una sola rutina que se ocupe de ello dentro del programa de trasformación. La multiplicación de matrices no es una tarea complicada, pero hay que tener cuidado y efectuar las multiplicaciones en el orden adecuado: aunque el álgebra de la escuela nos enseñaba que A x B = B x A, en álgebra de matrices la multiplicación A x B no tiene por qué dar el mismo producto que B x A. En el Apéndice 2 pueden encontrarse las reglas de multiplicación de matrices. A continuación se ofrece la rutina para multiplicar dos matrices de 3 x 3. Observe que es bastante diferente de las multiplicaciones de matrices que ya habíamos encontrado en TRV2. Ello se debe a que las multiplicaciones se realizan en este caso entre la matriz de transformación y un solo dato coordenado. De nuevo, remitimos al Apéndice 2 para mayor aclaración de este extremo, si desea enterarse de qué está sucediendo aquí.

```
3000  rem      rutina de multiplicación de matrices
3010   rem     las dos matrices algebraicas deben estar contenidas en
               matrices de BASIC dimensionadas como a(3,3) y b(3,3)
3020  for i=1 to 3
3030  for j=1 to 3:ab=0
3040  for k=1 to 3
3050  ab=ab+a(i,k)*b(k,j)
3060  next k:c(i,j)=ab:next j: next i
3070  return
```

Pongamos ahora esta rutina en nuestro programa TRV2 ampliado. Desgraciadamente, no basta insertar la rutina en TRV2, ya que es preciso efectuar una serie de multiplicaciones; como veremos en la nueva versión, nos encontramos con ciertas complicaciones. El resultado final es determinar un superproducto de matrices que se utilizará para calcular las coordenadas definitivas. La versión final del programa de transformación (TRV3) contiene también una sección de lectura de un fichero que permite introducir los datos desde un fichero creado por SKETCH o FICHERO2D.

Programa TRV3

```
10 REM ****PROGRAMA TRANSFORMACION V3****
20 REM Efectua transformaciones en 2D de datos de un fichero creado con
SKETCH o FICH2D
30 CLS
40 INPUT "NOMBRE DE FICHERO";h$
45 OPENIN h$
50 INPUT #9,npts
55 DIM a(3,3),b(3,3),c(3,3)
60 DIM x(npts),y(npts),xp(npts),yp(npts)
65 xl=640:xh=0:yl=400:yh=0
70 FOR i=1 TO npts
75   INPUT #9,x(i),y(i)
```

```
80 REM clasifica para encontrar los valores maximos y minimos
85     IF x(i)<xl THEN xl=x(i)
90     IF x(i)>xh THEN xh=x(i)
95     IF y(i)<yl THEN yl=y(i)
100    IF y(i)>yh THEN yh=y(i)
110 NEXT i
120    INPUT #9,li
125 DIM ln(2,li)
130    FOR i= 1 TO li
135      INPUT #9,ln(1,i),ln(2,i)
140    NEXT i
150 CLOSEIN
160 REM entrada de los datos relativos a la transformacion
165    INPUT "Traslacion TX,TY";tx,ty
170    INPUT "Cuantia de los cambios de escala";sx,sy
180    INPUT "Rotacion en grados";an
190 REM ahora pone los datos coordenados en matrices temporales
192    FOR i=1 TO npts
194      xp(i)=x(i)
196      yp(i)=y(i)
198   NEXT i
200 GOSUB 2500:REM dibujo de la figura original
210 REM ahora hace las transformaciones
220  t1=-((xh+xl)/2):t2=-((yh+yl)/2):REM  traslada  al  origen  el  punto
central
230 REM define las matrices y efectua las multiplicaciones
275 GOSUB 1400:REM rotacion (Matriz A)
280 GOSUB 1100:REM Traslacion al origen (Matriz B)
300 GOSUB 1600:REM Primera multiplicacion (Matriz C)
310 GOSUB 1500:REM Rotacion (Matriz B)
315 GOSUB 1300:REM Cambio de escala (Matriz B)
320 GOSUB 1700:REM Segunda multiplicacion (Matriz A)
330 t1=-t1+tx:t2=-t2+ty:REM Ajuste de los valores de traslacion
340 GOSUB 1100:REM Traslacion (Matriz B)
350 GOSUB 1600:REM Tercera multiplicacion (Matriz C)
360 GOSUB 2000:REM Calculo de las coordenadas
370 GOSUB 2500:REM Dibujo de la figura
380 INPUT x:IF x=1 THEN CLS:GOTO 160
400 END
1000 REM Rutina de traslacion
1010     a(1,1)=1:a(1,2)=0:a(1,3)=t1
1020     a(2,1)=0:a(2,2)=1:a(2,3)=t2
1030     a(3,1)=0:a(3,2)=0:a(3,3)=1
1040 RETURN
1100 REM Rutina de traslacion
1110     b(1,1)=1:b(1,2)=0:b(1,3)=t1
1120     b(2,1)=0:b(2,2)=1:b(2,3)=t2
1130     b(3,1)=0:b(3,2)=0:b(3,3)=1
1140 RETURN
1250 REM Rutina de cambio de escala
```

```
1260     a(1,1)=sx:a(1,2)=0:a(1,3)=0
1270     a(2,1)=0:a(2,2)=sy:a(2,3)=0
1280     a(3,1)=0:a(3,2)=0:a(3,3)=1
1290 RETURN
1300 REM Rutina de cambio de escala
1310     b(1,1)=sx:b(1,2)=0:b(1,3)=0
1320     b(2,1)=0:a(2,2)=sy:a(2,3)=0
1330     b(3,1)=0:b(3,2)=0:b(3,3)=1
1340 RETURN
1400 REM Rutina de rotacion
1410     a(1,1)=COS(an):a(1,2)=SIN(an):a(1,3)=0
1420     a(2,1)=-SIN(an):a(2,2)=COS(an):a(2,3)=0
1430     a(3,1)=0:a(3,2)=0:a(3,3)=1
1500 REM Rutina de rotacion
1510     b(1,1)=COS(an):b(1,2)=SIN(an):b(1,3)=0
1520     b(2,3)=-SIN(an):b(2,2)=COS(an):b(2,3)=0
1530     b(3,1)=0:b(3,2)=0:b(3,3)=1
1540 RETURN
1600 REM  Multiplicacion de las matrices (el resultado queda en C)
1610     FOR i=1 TO 3
1620     FOR j=1 TO 3:ab=0
1630     FOR k=1 TO 3
1640       ab=ab+a(i,k)*b(k,j)
1650     NEXT k
1660      c(i,j)=ab
1670     NEXT j
1680     NEXT i
1690 RETURN
1700 REM  RUTINA MULTIPLICADORA DE MATRICES
1710     FOR i=1 TO 3
1720     FOR j=1 TO 3:ab=0
1730     FOR k=1 TO 3
1740     ab=ab+b(i,k)*c(k,j)
1750     NEXT k
1760      a(i,j)=ab
1770     NEXT j
1780     NEXT i
1790 RETURN
2000 REM procesamiento de los puntos de la figura
2010  FOR i=1 TO npts
2020   p(1)=0:p(2)=0:p(3)=0
2030   po(1)=xp(i):po(2)=yp(i):po(3)=1
2040   FOR j=1 TO 3
2050   FOR k=1 TO 3
2060     p(j)=(c(j,k)*po(k))+p(j)
2070   NEXT k
2080   NEXT j
2090     xp(i)=p(1)
2100     yp(i)=p(2)
```

```
2110    NEXT i
2500 REM ahora dibuja la figura
2510    FOR i=1 TO li
2520       MOVE xp(ln(1,i)),yp(ln(1,i))
2530       DRAW xp(ln(2,i)),yp(ln(2,i))
2540    NEXT i
2550 FOR i=1 TO 1000:NEXT i:RETURN
```

Observando las líneas 1000-1450 de TRV3, puede verse que las matrices se generan con un procedimiento exactamente igual al utilizado en TRV2, pero en este caso las rutinas de las transformaciones están duplicadas para cada tipo de transformación. Esto es necesario, puesto que las multiplicaciones exigen que las matrices algebraicas estén en A y B, o en B y C. La matriz C es, en efecto, el producto resultante de la multiplicación de las matrices A y B, mientras que el producto de B y C se guarda en A.

La multiplicación de matrices que produce las coordenadas de dibujo (líneas 2000-2110) emplea la tabla C como matriz de transformación final. Hay que tener presente esto al ampliar TRV3.

Puede intentar algunas variaciones en el conjunto de los datos y las secuencias de transformaciones para asegurarse de que ha adquirido el dominio suficiente para manejar estas manipulaciones de matrices. Manipulando algunos trazados complejos con SKETCH podremos calibrar la velocidad a la que funcionan las diferentes transformaciones en BASIC.

4.6 Ventanas al mundo

Hemos de introducir ahora dos términos muy frecuentes en los artículos y libros acerca de los gráficos por ordenador: *ventanas* y *visores*. En el diagrama que se ofrece a continuación se muestra cómo están relacionados. Si pensamos en una escena de dos dimensiones como el "mundo" bidimensional que se desea visualizar, la ventana será la parte de ese mundo que vamos a ver en la pantalla.

Para la mayoría de las aplicaciones más sencillas, puede haber una correspondencia biunívoca entre la ventana y el mundo - los trazados que hemos venido realizando hasta ahora entran dentro de esta categoría. Sin embargo, otras representaciones exigen visualizar distintas partes del mundo. Es preciso, por tanto, definir una ventana que pueda colocarse en cualquier punto del mundo de 2 dimensiones. Consideremos, por ejemplo, un mapa, o el esquema de un circuito. La cantidad total de información contenida en las imágenes de este tipo será mucho mayor que la que podría visualizarse en una sola pantalla al mismo tiempo. El empleo de las ventanas permite visualizar por separado cualquier subconjunto de la escena total. Puede interesar, por ejemplo, desplazar la ventana sobre el mapa. Disponiendo de esta capacidad y de la técnica de ventanas, junto con la serie de datos adecuados para estos procedimientos, esto será posible.

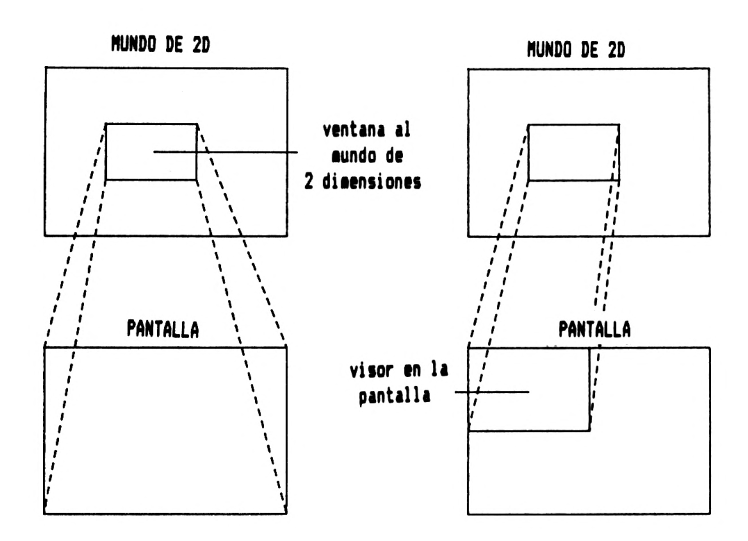


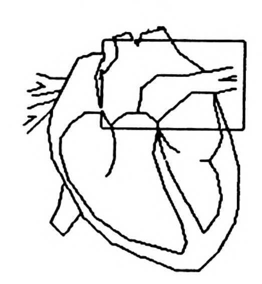
Figura 4.5 Relación entre el mundo de 2 dimensiones, las ventanas y los visores. Una relación 1:1 entre el mundo de 2D y la pantalla dará a la misma las proporciones del mundo bidimensional, y al visor el tamaño de la pantalla.

El visor es simplemente la parte de la pantalla en la que se visualiza la ventana. Los visores son útiles si se van a utilizar regiones distintas de la pantalla para presentar cosas diferentes: un ejemplo de ello es el caso en que parte de la pantalla se va a emplear para los gráficos y parte para el texto. La capacidad de separar de esta manera el texto y los gráficos es una particularidad de los ordenadores Amstrad, por medio de la orden WINDOW. Debe advertirse que WINDOW sólo es capaz de manipular por separado canales de texto; los comandos gráficos no están asociados con los canales, por lo que no pueden manejarse en una ventana por el BASIC de Amstrad. Combinando las técnicas de ventanas y visores, pueden simularse efectos panorámicos y de "zoom". Observe que se cumplen las siguientes reglas:

(1) Si disminuye el tamaño de la ventana, manteniendo constante el del visor, se produce un "zoom" acercándose.
(2) Si aumenta el tamaño de la ventana y el visor permanece constante, el efecto es de "zoom" alejándose.
(3) Si la ventana se mueve por el mundo de dos dimensiones se produce el efecto de "barrido" de la escena.

El tamaño de la ventana puede alterarse dinámicamente durante la ejecución de ZOOM, pulsando la tecla S (que encoge la ventana) y la tecla

L (que la amplía). El joystick se emplea para desplazarse por la superficie de la figura visualizada, y, cuando la ventana está en la posición apropiada y tiene el tamaño correcto, pulsando el botón de disparo se lleva a cabo la operación de "zoom". Para volver a la figura original, hay que pulsar de nuevo el botón de disparo, seguido de la tecla N. Puede emplearse el programa ZOOM para visualizar un espacio de dos dimensiones mucho mayor que el espacio coordenado de 640 x 400, y a la inversa, para expandir una figura más pequeña de forma que ocupe toda la pantalla. He aquí un ejemplo de dibujo obtenido con el programa ZOOM.



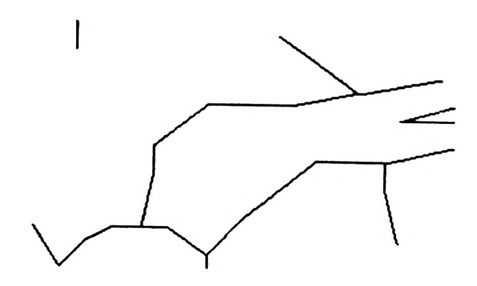


Figura 4.6 Figuras obtenidas con ZOOM: un corazón humano. Observe que la posición de la ventana del cursor en la figura superior define el área cubierta por la segunda figura.

Además de la traslación y el cambio de escala (operaciones en las que ya debe ser Vd. experto), ZOOM emplea el concepto de *truncamiento.* Esto significa que las líneas que atraviesan los límites de la ventana escogida se "cortan", de este modo:

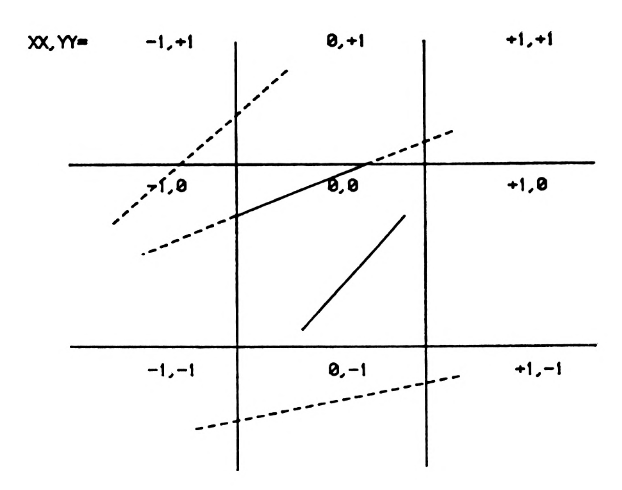


Figura 4.7 Operación de truncamiento. La pantalla se divide en nueve rectángulos, estando el espacio visible en el rectángulo central. Observe que los valores de XX e YY son iguales a 1 si las coordenadas son mayores que las del espacio visible, 0 si están dentro del mismo y -1 si son menores que las del espacio visible.

El algoritmo empleado para ello es un procedimiento estándar inventado hace algunos años por un pionero de los gráficos, el americano Ivan Sutherland, y consiste en ir moviendo por los límites de la ventana todos los finales de las líneas que salen de la pantalla. El algoritmo de truncamiento se encuentra codificado entre las líneas 160-1540 de ZOOM, y funciona como sigue: si se observa el diagrama anterior, podrá advertirse que toda el área se divide en nueve rectángulos, y la posición de las líneas horizontales y verticales viene determinada por la posición, trazado y tamaño de la ventana a manejar.

Todo punto de la pantalla se encontrará en uno de los nueve rectángulos, y utilizaremos esa información como base del algoritmo. Como puede verse en el diagrama, los rectángulos se identifican por los valores de dos variables, XX e YY. Si XX e YY son ambas 0 para los dos puntos de la línea, sabremos que toda ella será visible dentro de la ventana. Si XX es igual a -1 ó +1 entonces, sea cual sea el valor de YY, la línea quedará completamente fuera de la ventana. Si YY es igual a -1 ó +1, la línea también quedará totalmente fuera de la ventana. En los otros casos existe la posibilidad (no la seguridad) de que parte de la línea esté dentro de la ventana, y uno o quizá ambos puntos extremos de la línea se encuentren fuera.

En estos casos, la posición del punto exterior a la ventana se proyectará hasta la frontera de la misma (por ejemplo, c' en el diagrama anterior).

El algoritmo utiliza dos rutinas: TIPO devuelve los valores de XX e YY para identificar el rectángulo a que pertenece cada punto de la figura. PROYECCION emplea esta información para desplazar los puntos del exterior de la ventana a la frontera de la misma cuando es necesario.

Programa ZOOM

```
5 REM****PROGRAMA ZOOM****
10 REM toma una figura en formato FICHER2D y expande la seccion demarcada por la ventana
20 REM define las variables I,J,K,L,R como enteras
30 DEFINT i,j,k,l,r
40 CLS:MODE 2
45 GOSUB 1500:REM titulos
50 LOCATE 1,1:INPUT "NOMBRE DE FICHERO";h$
60 OPENIN h$
70   INPUT #9,npts
80   DIM x(npts),y(npts)
90   FOR i=1 TO npts
100    INPUT #9,x(i),y(i)
110  NEXT i
120  INPUT #9,li
130  DIM ln(2,li)
140  FOR i=1 TO li
150    INPUT #9,ln(1,i),ln(2,i)
160  NEXT i
170 CLOSEIN
180 REM entrada de la informacion de demarcacion y corte
190   xw=640:yw=400
200   k$="N"
210   xc=320:yc=200:REM define el punto central
220   zx=1:zy=1
230   wx=40:wy=INT(40*0.625):x=320:y=200:ww=5
240   dx=xw/2:dy=yw/2
250 REM seccion de truncamiento
260   xm=(640*zx)/xw
270   ym=(400*zy)/yw
280 REM escribe el titulo
300   FOR i=1 TO li
310     x1=x(ln(1,i))-xc
320     y1=y(ln(1,i))-yc
330     x2=x(ln(2,i))-xc
340     y2=y(ln(2,i))-yc
350   IF k$="N" OR k$="n" THEN xm=1:GOTO 660
360   xt=x1:yt=y1:GOSUB 860:REM modo truncamiento
370   i1=r1:i2=r2
380   xt=x2:yt=y2:GOSUB 860:REM modo truncamiento
390   i3=r1:i4=r2
400 REM estan todos los puntos fuera de la ventana?
410   IF (i1*i3=1) OR (i2*i4=1) THEN 710
```

```
420   IF i1=0 THEN 490
430 REM mueve la coordenada x del punto 1 a la frontera de la pantalla
440   xx=dx*i1
450   y1=y1+(y2-y1)*(xx-x1)/(x2-x1)
460   x1=xx
470   xt=x1:yt=y1:GOSUB 860:REM modo truncamiento
480   i1=r1:i2=r2
490   IF i2=0 THEN 540
500 REM mueve la coordenada y del punto 1 a la frontera de la pantalla
510   yy=dy*i2
520   x1=x1+(x2-x1)*(yy-y1)/(y2-y1)
530   y1=yy
540   IF i3=0 THEN 590
550 REM mueve la coordenada x del punto 2 a la frontera de la pantalla
560   xx=dx*i3
570   y2=y1+(y2-y1)*(xx-x1)/(x2-x1)
580   x2=xx
590   xt=x2:yt=y2:GOSUB 860:REM
600   i3=r1:i4=r2
610   IF i4=0 THEN 660
620 REM mueve la coordenada y del punto 2 a la frontera de la pantalla
630   yy=dy*i4
640   x2=x1+(x2-x1)*(yy-y1)/(y2-y1)
650   y2=yy
660 REM ya estamos preparados para dibujar
670      x1=(x1*xm)+320:x2=(x2*xm)+320
680      y1=(y1*ym)+200:y2=(y2*ym)+200
690         MOVE x1,y1
700         DRAW x2,y2
710   NEXT i
720 REM ahora se maneja la ventana
730      GOSUB 960:REM
740      GOSUB 1030:REM
750      GOSUB 1160:REM
760 IF JOY(0)<>16 THEN 720
765 CLS:GOSUB 1500
770      xw=x2-x1:yw=y2-y1:xc=x:yc=y
775 GOTO 240
850 END
860 REM subrutina del modo truncamiento
870      r1=0:r2=0
880      IF ABS(xt)>dx THEN r1=1:IF xt<0 THEN r1=-1
890      IF ABS(yt)>dy THEN r2=1:IF yt<0 THEN r2=-1
900 RETURN
950 REM rutina de dibujo de la ventana
960      x1=x-wx:y1=y-wy:x2=x+wx:y2=y+wy
970         MOVE x1,y1
980         DRAW x2,y1,1,1
990         DRAW x2,y2,1,1
1000        DRAW x1,y2,1,1
```

```
1010        DRAW x1,y1,1,1
1020 RETURN
1030 REM rutina de movimiento de la ventana
1040   y3=y:x3=x
1050        IF JOY(0)=4 THEN x=x-ww:GOTO 1090
1060        IF JOY(0)=8 THEN x=x+ww:GOTO 1090
1070        IF JOY(0)=1 THEN y=y+ww:GOTO 1090
1080        IF JOY(0)=2 THEN y=y-ww:GOTO 1090
1090          MOVE x1,y1
1100          DRAW x2,y1,1,1
1110          DRAW x2,y2,1,1
1120          DRAW x1,y2,1,1
1130          DRAW x1,x1,1,1
1140 RETURN
1150 REM rutina que bascula el tamaño de la ventana
1160 k$=INKEY$:IF k$="S" OR k$="s" THEN wx=wx-2:wy=wy-1.25
1170 IF k$="L" OR k$="l" THEN wx=wx+2:wy=wy+1.25
1172 IF k$="N" OR k$="n" THEN CLS:GOSUB 1500:GOTO 180
1180           MOVE x1,y1
1190           DRAW x2,y1,1,1
1200           DRAW x2,y2,1,1
1210           DRAW x1,y2,1,1
1220           DRAW x1,y1,1,1
1230           DRAW x2,y1,1,1
1240           DRAW x2,y2,1,1
1250           DRAW x1,y2,1,1
1260           DRAW x1,y1,1,1
1270 RETURN
1500 REM titulos
1510 LOCATE 34,1:PRINT "PROGRAMA ZOOM"
1520  LOCATE  1,25:PRINT"    L=AMPLIAR  VENTANA    S=COMPRIMIR  VENTANA
DISPARO=ZOOM  N=FIGURA"
1530 MOVE 0,30
1540 DRAW 640,30
1550 RETURN
```

Estas son las principales secciones de ZOOM:

LINEAS	
5	TITULO
20- 45	PREPARACION DE LA PANTALLA Y DECLARACION DE VARIABLES
50- 170	ENTRADA DE DATOS
190-	DEFINICION DEL TAMAÑO INICIAL DE LA VENTANA
210	DEFINICION DEL CENTRO DE LA VENTANA
230- 240	DEFINICION DEL TAMAÑO DE LA VENTANA DEL CURSOR, LA POSICION Y EL INCREMENTO EN LOS MOVIMIENTOS
250- 650	SECCION DE TRUNCAMIENTO
660- 710	DIBUJO DE LA FIGURA
720- 760	BUCLE HASTA QUE LA VENTANA DEL CURSOR CAMBIE
860- 900	SUBRUTINA DE MUESTREO DE SECTORES

950-1020	SUBRUTINA DE DIBUJO DE LA VENTANA DEL CURSOR
1030-1140	MOVIMIENTO DE LA VENTANA DEL CURSOR
1150-1270	CONMUTACION DEL TAMAÑO DE LA VENTANA DEL CURSOR

Los datos para ZOOM se leen de cualquier fichero de datos de 2 dimensiones que utilice el formato estándar que ya hemos empleado en SKETCH y FICHERO2D. Tenemos así un útil procedimiento para ampliar partes de un conjunto complejo de datos. Sin embargo, si los datos originales se han creado en pantalla mediante SKETCH, el valor de ZOOM presenta algunas limitaciones. El efecto de ZOOM es mucho más espectacular si somos capaces de "caminar" por una figura constituida por datos dibujados originalmente en un tamaño varias veces superior al de la pantalla. Hay varias formas de generar datos mayores que la pantalla, ya sea con FICHERO2D o modificando SKETCH. La siguiente versión de SKETCH (llamada CUADRANTE) permite crear una figura de tamaño cuatro veces mayor que la pantalla mediante la técnica de SKETCH. Como su nombre sugiere, CUADRANTE va definiendo los segmentos sucesivos como cuadrantes de la figura final. El cuadrante superior izquierdo es el primer segmento, el superior izquierdo es el segundo, el inferior izquierdo el tercero y el inferior derecho el cuarto. La figura definitiva queda, pues, como se ve en la figura 4.8.

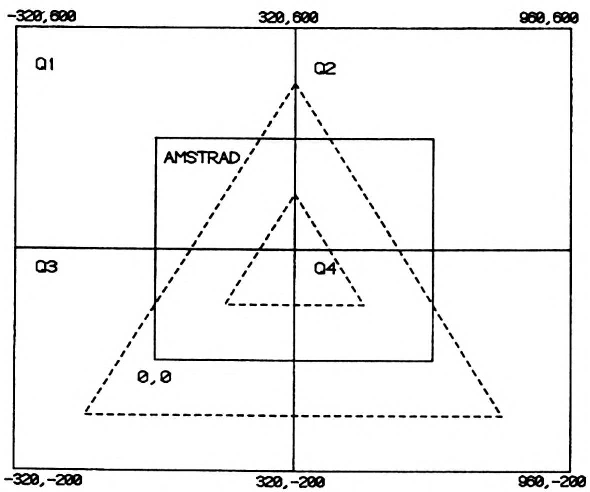


Figura 4.8 Los cuatro cuadrantes definidos por CUADRANTE. Observe que la ventana se extiende desde -320,200 hasta 960,600. La siguiente transformación ajusta la escala de los puntos dibujados en este área a las dimensiones del sistema normal de coordenadas de la pantalla del Amstrad

COORD X = (320 + X)/2
COORD Y = (200 + Y)/2

Los triángulos a trazos muestran el efecto del cambio de escala.

Para pasar de SKETCH a CUADRANTE es necesario hacer las siguientes modificaciones:

Programa CUADRANTE

```
1 REM****AMPLIÁCION PARA CUADRANTES DE SKETCH****
2 REM para añadir estas lineas cargue primero SKETCH y luego mezcle
CUADRANTE"
3 qu=1:REM ajusta el contador de cuadrantes
135 GOSUB 2700:REM cambia la visualizacion de texto para el cuadrante
750 REM sobreescribir esta linea
755 GOSUB 2500:REM ajusta las coordenadas del segmento
757 REM sobreescribir esta linea
770 IF qu=5 THEN 800
2100 REM rutina de ajuste de la frontera
2110     IF x<0 THEN x=0
2120     IF x>640 THEN x=640
2130     IF y<0 THEN y=0
2140     IF y>400 THEN y=400
2150     IF xi<0 THEN xi=0
2160     IF xi>640 THEN xi=640
2170     IF yi<0 THEN yi=0
2180     IF yi>400 THEN yi= 400
2190 RETURN
2500 REM rutina de ajuste de las coordenadas del cuadrante
2510 i1=ln(1,s(1,qu)):i2=ln(2,s(2,qu))
2520 FOR i=i1 TO i2
2530   IF qu=1 THEN xp(i)=xp(i)-320:yp(i)=yp(i)+200:GOTO 2600
2540   IF qu=2 THEN xp(i)=xp(i)+320:yp(i)=yp(i)+200:GOTO 2600
2550   IF qu=3 THEN xp(i)=xp(i)-320:yp(i)=yp(i)-200:GOTO 2600
2560   IF qu=4 THEN xp(i)=xp(i)+320:yp(i)=yp(i)-200:GOTO 2600
2600 NEXT i
2610 qu=qu+1
2620 IF qu=1 THEN GOTO 800
2630 GOSUB 2700
2640 RETURN
2700 REM texto del numero de cuadrante
2702 CLS
2705 LOCATE 12,1:PRINT "PROGRAMA CUADRANTE"
2710 LOCATE 1,2
2720 PRINT "CUADRANTE NUMERO ";qu
2730 LOCATE 1,23
2740 PRINT "PULSE E PARA TERMINAR EL CUADRANTE"
2800 REM rutina localizadora de la dirección
2805 IF qu=1 THEN x9=47:y9=298
2810 IF qu=2 THEN x9=563:y9=298
2815 IF qu=3 THEN x9=47:y9=99
2820 IF qu=4 THEN x9=563:y9=99
2825 REM ahora dibuja los cuatro cuadrados
2830   FOR i=1 TO 4
```

```
2835      IF i=2 THEN x9=x9+25
2840      IF i=3 THEN y9=y9+25
2845      IF i=4 THEN x9=x9-25
2850      MOVE x9,y9:REM se desplaza a la posición relativa de comienzo
2855      DRAWR 20,0
2860      DRAWR 0,-20
2865      DRAWR -20,0
2870      DRAWR 0,20
2875 NEXT i
2880      IF qu=1 THEN LOCATE 4,6:PRINT"1"
2885      IF qu=2 THEN LOCATE 38,6:PRINT"2"
2890      IF qu=3 THEN LOCATE 4,20:PRINT"3"
2895      IF qu=4 THEN LOCATE 38,20:PRINT"4"
2900 REM rutina de dibujo de la cuadricula
2905 REM LA ORDEN MASK PARA DIBUJAR LAS LINEAS DE PUNTOS SOLO
FUNCIONARA CON EL BASIC DEL CPC 664/6128
2910 FOR i=0 TO 640 STEP 40
2920 MOVE i,0
2930 MASK 36:DRAW i,400
2950 NEXT i
2955 FOR i=0 TO 400 STEP 40
2960 MOVE 0,i
2965 MASK 36:DRAW 640,i
2970 NEXT i
2975 MASK 255
2980 RETURN
```

Los datos generados por CUADRANTE están dentro del margen de valores desde -320,-200 hasta 960,600, lo cual nos da una "resolución" de 1280 x 800 pixels, aunque, por supuesto, no todos ellos serán visibles directamente en la pantalla del Amstrad. Para manejar estos datos se emplea una versión mejorada de ZOOM (llamada ZOOMCUAD). A continuación se indican las modificaciones necesarias. Operacionalmente, ZOOMCUAD es idéntico a ZOOM, por lo que las teclas de control y el joystick se emplean igual en ambas versiones.

Las modificaciones que hay que efectuar en ZOOM para obtener ZOOMCUAD son bastante menos severas que las necesarias para producir CUADRANTE. Estas son:

```
10 REM *** VERSION ZOOMCUAD DE ZOOM ***
105 X(I)=(X(I)+320)/2:Y(I)=(Y(I)+200)/2
```

Hemos utilizado CUADRANTE para crear el mapa de los Estados Unidos que se muestra en los siguientes volcados de pantalla de ZOOMCUAD.

NOMBRE DEL FICHERO? USA,DAT PROGRAMA ZOOM



L=AMPLIAR VENTANA S=REDUCIR VENTANA DISPARO=ZOOM N=FIGURA DE TAMAÑO NORMAL

Figura 4,9 Resultado de ZOOMCUAD: mapa de los Estados Unidos definido mediante CUADRANTE en cuatro segmentos separados (es decir, NO, NE, SE y SO): observe la "ventana" del cursor.

PROGRAMA ZOOM

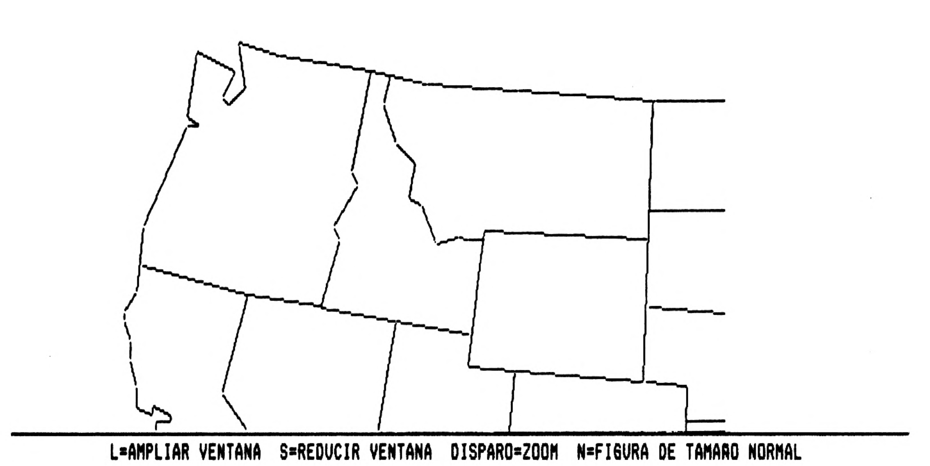


Figura 4,10 Detalle de una zona del mismo mapa,

En este capítulo hemos visto bastantes fundamentos de las técnicas gráficas en dos dimensiones. Ha llegado el momento de agruparlos para producir un programa "útil". Estamos limitados en cuanto a posibilidades por el BASIC, ya que nuestro pobre intérprete de BASIC simplemente no puede trabajar con la rapidez suficiente para multiplicar en tiempo real largas series de matrices. Esta limitación se vuelve absoluta si se desea programar juegos de movimiento rápido, o en el caso de un simulador de vuelo. Hemos visto ya que podemos emplear un método para abreviar las rotaciones en dos dimensiones, y en los capítulos relativos a los gráficos tridimensionales examinaremos otras formas de superar las limitaciones de velocidad. Pero no hay que desanimarse del todo. Cualesquiera que sean los motivos que le impulsaron a aprender las técnicas gráficas, no podrá ir muy lejos sin un arsenal de herramientas como el que este capítulo pone a su disposición. Incluso aunque vaya a programar en código máquina, los algoritmos seguirán siendo, en esencia, los mismos.

Tal vez suspire decepcionado porque lo que Vd. quería era aferrarse al BASIC. No se preocupe. No todo está perdido. Hay muchas aplicaciones aparte de los juegos frenéticos y vertiginosos que pueden no necesitar efectos de animación. La mayor parte del Capítulo 6 se ocupará de describir una aplicación de este tipo: un programa de diseño asistido por ordenador. Este programa muestra una aplicación, pensamos, no precisamente trivial de las rutinas presentadas en este capítulo.

Capítulo 5

G r á f i c o s c o m e r c i a l e s

5.1 La importancia de la presentación

Los gráficos por ordenador se emplean extensivamente en el campo de los negocios, como ayuda a los estudios de mercado, análisis financieros y planificaciones. Los gráficos se utilizan para producir diagramas y gráficas que muestren datos como las tendencias comerciales, ventas comparativas, fluctuaciones presupuestarias, *cash-flow*... La información presentada en forma gráfica se entiende con mayor rapidez que las correspondientes páginas de tablas, y el impacto de una imagen es mucho mayor que el de los datos más abstractos.

Tres son los tipos más importantes de técnicas de ilustración para aplicaciones comerciales. El primero de ellos son las gráficas, de las cuales podemos ver un ejemplo en el Capítulo 1. Son especialmente adecuadas para presentar datos precisos, por ejemplo, si es necesario ilustrar un gran número de puntos. Una variación sobre este mismo tema son los histogramas o diagramas de barras, en los cuales se presentan los datos en pasos, proporcionando un aspecto más atractivo a la representación. La tercera técnica gráfica comercial son las gráficas en tarta, en las que las proporciones relativas del total (es decir, los trozos de la tarta) se ilustran con diferentes sectores circulares.

Los paquetes para gráficos comerciales disponibles en el mercado suelen ofrecer representaciones muy atractivas y complejas, mediante sofisticados programas. Puede ser posible incluso visualizar en pantalla distintos tratamientos de los mismos datos (por ejemplo, una gráfica y un histograma) al mismo tiempo. Afortunadamente para el usuario de ordenadores domésticos, las técnicas empleadas en los gráficos comerciales suelen ser en realidad muy simples. La complejidad de los programas suele deberse a la gran cantidad de capturas de errores de que disponen estos programas para hacer más agradable su manejo. El usuario de un Amstrad posee además otra ventaja sustancial. Su pantalla dispone de un modo de 80 columnas que permite producir gráficos comerciales de aspecto mucho más profesional que los que podrían generarse con la mayoría de los micros domésticos. Así pues, ¿qué es lo que podemos hacer en nuestro Amstrad?

5.2 Un trozo de la tarta

Ya hemos visto en el programa CIRCULO un algoritmo para la generación de circunferencias. La generación de círculos es la parte más destacada de

un programa de gráficas tipo tarta. Lo más importante a tener en cuenta acerca de los datos de las gráficas en tarta es que es preciso expresarlas en términos de proporciones del total. Podemos emplear una tarta, por ejemplo, para representar las ventas relativas de palomitas en diferentes áreas de mercado, pero no puede utilizarse este método para ilustrar las ventas durante un periodo de doce meses. Los datos en bruto para una gráfica de tarta como ésta deberán ser algo similar a esto:

Zona		Ventas de palomitas
Centro		2.530.000
Suroeste		924.000
Levante		1.346.000
Norte		2.980.000
Cataluña		3.250.000
	Total	10.893.000

Estos datos nos informan de varias cosas. En primer lugar, la tarta tendrá cinco sectores (uno para cada área). Segundo, las ventas regionales se representarán todas en la gráfica como proporción de las ventas totales. Es más sencillo pensar en términos de porcentaje; así, el porcentaje de ventas en Cataluña resultará ser

$$\frac{3.250.000}{10.830.000} \times \frac{100}{1}$$

es decir, un 30 por ciento aproximadamente.

La primera tarea que debe realizar un programa de gráfica de tarta es la entrada de los datos, su totalización y la determinación de las proporciones correspondientes a cada elemento del total. Pero , ¿cómo hacer corresponder a los sectores radiales los porcentajes brutos? Un gráfico de tarta consiste en una serie de líneas que irradian de un punto central, y el sector comprendido entre dos líneas adyacentes representa la proporción de un elemento determinado. El ángulo entre cada par de líneas es, pues, crucial. En lugar de 100 puntos porcentuales para la suma de todos los datos, tenemos los 360 grados de un círculo completo. Si tomamos el porcentaje de cualquier elemento, el ángulo comprendido entre las líneas que representan sus límites en la gráfica de tarta será:

$$\frac{\text{porcentaje del elemento}}{100} \times 360 \text{ grados}$$

La siguiente tarea que debe realizar el programa de la tarta es, por lo tanto, calcular los equivalentes angulares de los datos porcentuales. La única técnica fundamental que es necesario escribir es un método para calcular las líneas radiales. Utilizará las mismas ecuaciones empleadas para generar una circunferencia. Recuerde del programa CIRCULO que para

todo punto de la circunferencia las coordenadas X e Y vienen definidas por las ecuaciones

X = XC + (RADIO * COS (ANGULO))

Y = YC + (RADIO * SIN (ANGULO))

Donde XC e YC son las coordenadas del centro de la circunferencia.

Así, los segmentos radiales que definen el primer sector de la tarta, suponiendo que el ángulo es conocido, estarán definidos por las líneas

```
MOVE XC,YC
DRAW X,Y      (donde ANGULO = 0)
MOVE XC,YC
DRAW X1,Y1    (donde ANGULO = proporción para el primer segmento)
```

Pero las tartas monocromáticas son muy aburridas. Tropezamos aquí con un problema tanto en el CPC 664 como en el CPC 464, puesto que el modo de 80 columnas sólo nos permite dos colores simultáneos en pantalla. Se nos plantea el compromiso entre un texto pequeño (y por consiguiente denso) pero sin color, o un texto rechoncho con cuatro colores en MODE 1. Si lo que desea en realidad es una tarta de colorines, utilice el MODE 0. Desgraciadamente, el texto en MODE 0 es tan ancho que deberá limitarse a etiquetas de una letra,o a lo sumo dos. La elección está en sus manos. El siguiente programa TARTA utiliza MODE 1, pero puede adaptarse fácilmente para los modos 0 ó 2.

Programa TARTA

```
10 REM **** PROGRAMA TARTA****
20 REM DIBUJA UNA GRAFICA DE TARTA CON ROTULOS EN CUATRO COLORES, EN
MODO 1
30 REM ENTRADA DE DATOS
40 CLS:INK 0,13:INK 1,0:INK 2,3:INK 3,7
50 MODE 1
60 PRINT"   Bienvenido al programa TARTA"
65 INPUT"TITULO PRINCIPAL";m$
66 p1=LEN(m$):p1=20-(p1/2)
70 INPUT"Cuantos segmentos deben visualizarse";numero
72 DIM s(numero),h$(numero),punto(numero),cum(numero)
73 total=0:cangulo=0
75 FOR i=1 TO numero
80 INPUT "TITULO DE ESTE SEGMENTO";h$(i)
85 INPUT "VALOR DEL SEGMENTO";s(i)
87 total=total+s(i)
90 NEXT i
100     FOR i=1 TO numero:REM ajusta los angulos para cada segmento
110     cangulo=cangulo+((s(i)/total)*(2*PI))
120     punto(i)=cangulo-(((s(i)/2)/total)*(2*PI))
```

```
122     cum(i)=cangulo
130     NEXT i
135 CLS
200 LOCATE p1,1:PRINT m$
205 TAG
220 REM ajusta el tamaño del circulo
230 radio=150
240 xc=320:yc=200
250         a=(2*PI)/100
260         angulo=0
270 x2=xc+radio:y2=yc
280   FOR i=1 TO 100
290     angulo=angulo+a
300     x1=x2:y1=y2
310     x2=xc+radio*COS(angulo)
320     y2=yc+radio*SIN(angulo)
330     MOVE x1,y1
340     DRAW x2,y2
350 NEXT i
400 REM ahora dibuja los segmentos
405 n=-1
410     FOR i=1 TO numero
415     n=n+1:IF n=4 THEN n=0
420     MOVE xc,yc
430     x1=xc+radio*COS(cum(i))
440     y1=yc+radio*SIN(cum(i))
450     DRAW x1,y1
460     x2=xc+(radio/2)*COS(punto(i))
470     y2=yc+(radio/2)*SIN(punto(i))
472     disp=LEN(h$(i))+15
475     MOVE x2,y2
480 FILL n
482     IF x2<xc THEN disp=disp+10
485     MOVE x2-disp,y2
487     PRINT h$(i);
490 NEXT i
```

Puede parecer muy fácil seguir la estructura de TARTA, pero es preciso advertir dos detalles. En primer lugar, el programa emplea la orden FILL del CPC 664/6128, por lo que los usuarios de CPC 464 no podrán colorear sus tartas. La segunda cuestión se refiere a la colocación del texto (líneas 460-487). El texto se sitúa en la pantalla gráfica empleando la orden TAG para enlazarlo a la posición del cursor gráfico. La posición normal de comienzo para cada cadena de texto se encuentra en la bisectriz del sector, a mitad de camino entre el centro del círculo y la circunferencia. Si el sector está en el lado izquierdo del círculo, se efectúa un desplazamiento de diez unidades hacia la izquierda.

Observe la forma de los datos en el programa. Los datos en bruto se introducen en la matriz S, mientras que el título del segmento se guarda

EL MERCADO AUTOMOVILISTICO EUROPEO - 1982

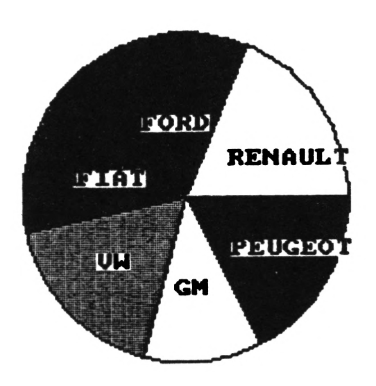


Figura 5.1 Gráfica de tarta creada con el programa TARTA, en modo 1.

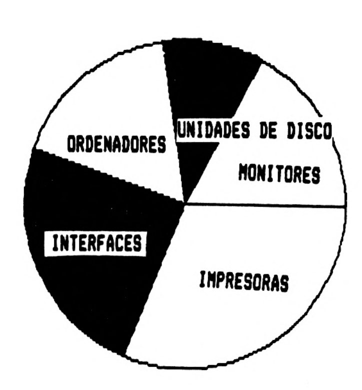


Figura 5.2 Gráfica de tarta creada con el programa TARTA, en modo 2

en la matriz H$. Los ángulos se calculan a continuación, y se introducen en una nueva matriz, CUM (abreviatura de aCUMulación), en la que se guarda el ángulo total para cada segmento DESDE 0 GRADOS. Si los ángulos de los tres primeros segmentos fuesen de 15 grados,15 grados y 5 grados,se representarían en la matriz CUM como 15, 30 y 35 grados, respectivamente. La matriz PUNTO guarda el ángulo de la bisectriz para el segmento actual, utilizada para la colocación de texto.

Existen diversas variaciones sobre el tema de las tartas. La primera de ellas es la tarta partida (Figura 5.3). Se emplea para enfatizar un segmento determinado de la tarta. La lógica del programa PARTICION que se muestra a continuación es similar a la del programa TARTA, pero son necesarias algunas secciones adicionales para manejar la sección destacada. En concreto, es necesario hallar la bisectriz del segmento separado, y el centro de la circunferencia debe desplazarse momentáneamente a lo largo de esa línea en la cantidad requerida.

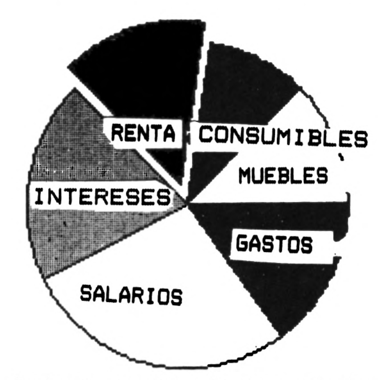


Figura 5.3 Gráfica de tarta con un segmento segregado (programa PARTICION)

Programa PARTICION

```
10 REM ****PROGRAMA PARTICION****
20 REM DIBUJA UNA TARTA CON ROTULOS EN CUATRO COLORES EN MODO 1
22 REM CON UNA SECCION SEGREGADA ESCOGIDA POR EL USUARIO
30 REM ENTRADA DE DATOS
40     CLS:INK 0,13:INK 1,0:INK 2,6:INK 3,12:MODE 1
60     PRINT"  Bienvenido al programa TARTA
70     INPUT"NUMERO DE SEGMENTOS A VISUALIZAR";numero
80     INPUT "NUMERO DEL SEGMENTO A SEGREGAR";particion
90 IF particion=numero THEN PRINT"NO PUEDE SEGREGARSE EL ULTIMO
SEGMENTO":PRINT"EMPIECE OTRA VEZ":GOTO 80
100 DIM s(numero),h$(numero),punto(numero),cum(numero)
110 total=0:cangulo=0
120     FOR i=1 TO numero
130        PRINT"ESCRIBA EL TITULO DEL SEGMENTO ";i:INPUT h$(i)
140        INPUT"VALOR DEL SEGMENTO";s(i)
```

```
150       total=total+s(i)
160     NEXT i
170 TAG
180     FOR i=1 TO numero:REM ajusta los angulos para cada segmento
190     cangulo=cangulo+((s(i)/total)*(2*PI))
200     punto(i)=cangulo-(((s(i)/2)/total)*(2*PI))
210     cum(i)=cangulo
220     NEXT i
230 CLS
240 REM ajusta el tamaño del circulo
250 radio=150
260    xc=320:yc=200
270       a=(2*PI)/300
280       angulo=0
290    x2=xc+radio:y2=yc
300      fl1=0:fl2=0:REM ajusta las banderas indicadoras del segmento
segregado
310    FOR i=0 TO 300
320      angulo=angulo+a
330      x1=x2:y1=y2
340      IF angulo>=cum(particion-1) AND fl1=0 THEN GOSUB 630:REM segrega
el segmento
350      IF angulo>=cum(particion) AND fl2=0 THEN MOVE xc,yc:DRAW x1,y1
360      IF angulo>=cum(particion) AND fl2=0 THEN xc=xhold:yc=yhold
370      x2=xc+radio*COS(angulo)
380      y2=yc+radio*SIN(angulo)
390      IF angulo>=cum(particion-1) AND fl1=0 THEN x1=x2:y1=y2
400         IF angulo>=cum(particion-1) AND fl1=0 THEN MOVE xc,yc:DRAW
x2,y2:fl1=1
410  IF angulo>=cum(particion) AND fl2=0 THEN angulo=angulo-a: x1=x2:
y1=y2: fl2=1:GOTO 330
420      MOVE x1,y1
430      DRAW x2,y2
440    NEXT i
450 REM ahora dibuja los segmentos
460 n=-1
470    FOR i=1 TO numero
480      n=n+1:IF n=4 THEN n=0
490      MOVE xc,yc
500      x1=xc+radio*COS(cum(i))
510      y1=yc+radio*SIN(cum(i))
520      DRAW x1,y1
530      x2=xc+(radio/2)*COS(punto(i))
540      y2=yc+(radio/2)*SIN(punto(i))
550      disp=LEN(h$(i))*3
560      MOVE x2,y2
570      FILL n
580      IF x2<xc THEN disp=disp*4
590      MOVE x2-disp,y2
600    NEXT i
```

```
610 GOSUB 700:REM Pone los títulos
615 |COPY:END
630 REM calcula las coordenadas del centro para el sector segregado
640 REM primero calcula la direccion del angulo
650 bis=((cum(particion)-cum(particion-1))/2)+cum(particion-1)
660    xhold=xc:yhold=yc:REM almacena los valores normales del centro
670    xc=xhold+20*COS(bis)
680    yc=yhold+20*SIN(bis)
690 RETURN
700 REM ahora dibuja los titulos
710    n=-1
720    FOR i=1 TO numero
730      n=n+1:IF n=4 THEN n=0
740      MOVE xc,yc
750      x1=xc+radio*COS(cum(i))
760      y1=yc+radio*SIN(cum(i))
770      DRAW x1,y1
780      x2=xc+(radio/2)*COS(punto(i))
790      y2=yc+(radio/2)*SIN(punto(i))
800      disp=LEN(h$(i))*3
810      MOVE x2,y2
820      IF x2<xc THEN disp=disp*4
830 MOVE x2-disp,y2
840      PRINT h$(i);
850    NEXT i
870 RETURN
```

La última variación es la visualización de varias "minitartas" al mismo tiempo para ilustrar datos más complejos, por ejemplo, las fluctuaciones proporcionales durante un periodo de tiempo. Al igual que sucedía con PARTICION, no existen verdaderas diferencias con el caso principal de TARTA, pero se plantean algunos problemas de colocación. En el ejemplo que se muestra a continuación (MINITARTA) se muestran 12 pequeñas tartas, que muestran las diferencias durante un periodo de 12 meses. Puede observarse que las doce tartas se dibujan utilizando el mismo bloque del programa (líneas 220-660), y basta una sola línea (la 320) para cambiar la posición de la tarta a dibujar. Teniendo en cuenta que se apiñan demasiados datos en la pantalla, éstos vienen incluidos en el propio programa, en lugar de introducirse de forma interactiva por el teclado. Por supuesto, puede emplearse un fichero de entrada de datos en cinta o disco si se desea. La concentración en pantalla obliga también a emplear etiquetas de una sola letra. Desgraciadamente, el modo 2 implica falta de color, pero la salida tiene un aspecto terrible en modo 1: ¡inténtelo y verá!

Programa MINITARTA

```
10 REM ****PROGRAMA MINITARTA****
20 REM Esta modificación dibuja 12 minitartas que sirven para comparar
30 REM los balances durante un periodo de 12 meses
```

```
40 REM Cada tarta se rotula con un código de una sola letra
45 REM Para una resolucion mas elevada, se utiliza el modo 2
50 REM Entrada de datos
60 CLS:INK 0,13:INK 1,0
70 MODE 2
80 PRINT" VENTAS RELATIVAS DE LAS LINEAS DE PRODUCTOS A-E DURANTE DOCE MESE"
90 READ numero
100 DIM s(numero,12),h$(numero),punto(numero,12),cum(numero,12),total(12)
110 total=0:cangulo=0
120   FOR i=1 TO numero
130   READ h$(i)
140   REM Los datos de cada mes estan al final del programa
150   REM num de segmentos, codigo del titulo, valores del segmento - para cada
uno
160   NEXT i
170   FOR i=1 TO 12
180     FOR ip=1 TO numero
190       READ s(ip,i):total(i)=total(i)+s(ip,i):NEXT ip:NEXT i
200   GOSUB 680:REM ahora escribe los meses
210   TAG
220 REM ahora dibuja las doce tartas
230   xc=0:yc=319:REM centro de la primera tarta
240   FOR ip=1 TO 12:REM comienzo del bucle para las doce tartas
250   FOR i=1 TO numero:REM ajusta los angulos para cada segmento
260   cangulo=cangulo+((s(i,ip)/total(ip))*(2*PI))
270   punto(i,ip)=cangulo-(((s(i,ip)/2)/total(ip))*(2*PI))
280   cum (i,ip)=cangulo
290   NEXT i
300 REM ajusta el tamaño del circulo
310   radio=45
320   xc=xc+127:IF xc>=605 THEN xc=125:yc=yc-110:REM empieza nueva fila
330   a=(2*PI)/100
340   angulo=0
350   x2=xc+radio:y2=yc
360   FOR i=1 TO 103
370     angulo=angulo+a
380     x1=x2:y1=y2
390     x2=xc+radio*COS(angulo)
400     y2=yc+radio*SIN(angulo)
410     MOVE x1,y1
420     DRAW x2,y2
430   NEXT i
440 REM ahora dibuja el segmento
450   n=-1
460   FOR i=1 TO numero
470     n=n+1:IF n=4 THEN n=0
480     MOVE xc,yc
490     x1=xc+radio*COS(cum(i,ip))
500     y1=yc+radio*SIN(cum(i,ip))
510     DRAW x1,y1
```

```
520       x2=xc+(radio/2)*COS(punto(i,ip))
530       y2=yc+(radio/2)*SIN(punto(i,ip))
540       MOVE x2,y2+6
550       IF INT(i/2)=i/2 THEN FILL 1
560     NEXT i
570 REM ahora dibuja los titulos
580  n=-1
590  FOR i=1 TO numero
600       n=n+1:IF n=4 THEN n=0
610       x1=xc+(radio/2)*COS(punto(i,ip))
620       y1=yc+(radio/2)*SIN(punto(i,ip))
630       MOVE x1,y1+6
640       PRINT h$(i);
650 NEXT i
660 NEXT ip
670 |COPY:END
680 REM dibuja el mes
690  LOCATE 9,9:PRINT"      ENE             FEB              MAR
700  LOCATE 9,16:PRINT"      MAY             JUN              JUL
AGO"
710 LOCATE 9,23:PRINT"      SEP             OCT              NOV
720 RETURN
730 DATA 5:REM
740 DATA "A","B","C","D","E"
750 DATA  20,20,20,20,20:REM ENE
760 DATA  30,10,15,40,12:REM FEB
770 DATA  10,15,30,20,10:REM MAR
780 DATA  12,12,34,12,10:REM ABR
790 DATA  14,31,12,10,19:REM MAY
800 DATA  15,31,12,8,19:REM JUN
810 DATA  13,13,14,15,12:REM JUL
820 DATA  22,15,10,12,18:REM AGO
830 DATA  15,13,10,13,17:REM SEP
840 DATA  14,41,13,24,13:REM OCT
850 DATA  5,12,13,15,13:REM NOV
860 DATA  11,21,31,8,13:REM DIC
```

VENTAS RELATIVAS DE LAS LINEAS DE PRODUCTOS
A-E DURANTE UN PERIODO DE 12 MESES

ENE FEB MAR ABR
MAY JUN JUL AGO
SEP OCT NOV DIC

Figura 5.4 Resultado del programa MINITARTA

5.3 Técnicas de representación gráfica

La primera forma de representación pictórica por ordenador fue probablemente la gráfica; todos nosotros estamos familiarizados con la abstracción de los números en la simple forma gráfica. Las gráficas funcionan con relaciones: muestran la variación de un parámetro con respecto a otro. En los casos más complejos, pueden dibujarse juntas tres o incluso cuatro variables para generar gráficas bastante complicadas. Nosotros nos ocuparemos únicamente de las gráficas de andar por casa, con dos ejes.

Después de estudiar el Capítulo 1, ya debe estar familiarizado con los métodos que intervienen en la programación de una simple gráfica. En la presente sección veremos cómo mejorar el programa elemental para hacerlo más cómodo y para que pueda producir una salida más útil. He aquí una versión mejorada de nuestro anterior programa GRAFICA, llamada SUPERG

Programa SUPERG

```
10 REM****PROGRAMA SUPERG****
20 REM Version del programa GRAFICA que representa dos variables
30 REM en una sola gráfica rotulada
40   INK 0,13
50   INK 1,0
60   READ titulo$
70 opcion=1:REM seleccione 1 para dibujar puntos y 2 para dibujar líneas
80 MODE 2
90 READ puntos
100 DIM x(puntos),y(puntos)
110   FOR i=1 TO puntos
120     READ x(i)
130     READ y(i)
140   NEXT i
150   READ xmin,xmax,ymin,ymax
160   READ x$:REM nombre del eje X
170   READ y$:REM nombre del eje Y
180 CLS
190 REM ahora dibuja los ejes
200   MOVE 100,380
210   DRAW 100,80
220   DRAW 550,80
230 REM genera las marcas de las escalas
240   FOR i=1 TO 11
250     MOVE 90,(i*30)+50
260     DRAW 100,(i*30)+50
270   NEXT i
280   FOR i=1 TO 16
290     MOVE (i*30)+70,70
300     DRAW (i*30)+70,80
310   NEXT i
```

```
320 REM imprime el titulo
330   p1=LEN(titulo$):p1=40-(p1/2)
340   LOCATE p1,1:PRINT titulo$;
350 REM ahora rotula los ejes
360 REM en primer lugar rotula el X
370 REM La posicion de comienzo es el centro del eje X menos la mitad de la
longitud de la cadena
380   ax=(320-((LEN(x$)*16)/2))
390 REM la posicion de comienzo es el centro del eje Y mas la mitad de la
longitud de la cadena
400   ay=(220+((LEN(y$)*16)/2))
410 TAG
420     MOVE ax,40
430   PRINT x$;
440 REM ahora imprime la leyenda del eje Y en vertical
450   FOR i=1 TO LEN(y$):m1$=MID$(y$,i,1)
460     MOVE 40,ay-((i-1)*16)
470     PRINT m1$;
480   NEXT i
490 MOVE 530,60:PRINT xmax;
500 MOVE 50,382:PRINT ymax;
510 MOVE 70,90:PRINT ymin;
520 MOVE 80,60:PRINT xmin;
530 MOVE 55,240:PRINT INT((ymax+ymin)/2);
540 MOVE 290,60:PRINT INT((xmax+xmin)/2);
550 REM ahora dibuja los puntos
560   IF opcion=2 THEN GOTO 640
570   FOR i=1 TO puntos
580     xtop=xmax-xmin:ytop=ymax-ymin
590     xtrue=xtop-(xmax-x(i)):ytrue=ytop-(ymax-y(i))
600     MOVE 96+(450*(xtrue/xtop)),86+(300*(ytrue/ytop))
610     PRINT CHR$(244);
620   NEXT i
630 |COPY:END
640 REM seccion de dibujo de lineas
650 FOR i=1 TO puntos
660   xtop=xmax-xmin:ytop=ymax-ymin
670   xtrue=xtop-(xmax-x(i)):ytrue=ytop-(ymax-y(i))
680   IF i=1 THEN MOVE 96+(450*(xtrue/xtop)),86+(300*(ytrue/ytop))
690   DRAW 96+(450*(xtrue/xtop)),86+(300*(ytrue/ytop))
700 NEXT i
710 |copy:END
720 REM secuencia de datos: TITULO, NUM DE PUNTOS, VALX, VALY DE CADA  PUNTO
730 REM                      XMIN, XMAX, YMIN, YMAX
740 REM                      X$, Y$
750 DATA "INDICE DE PARTICIPACION SOBRE UN PERIODO DE DIECISEIS AÑOS"
760 DATA 16,1970,380,1971,550,1972,500,1973,400,1974,270,1975,290,1976,480,1977
765 DATA 500,1978,500,1979,500,1980,495,1981,520,1982,540,1983,660,1984,800,1985,940
770 DATA 1970,1985,0,1000
780 DATA "AÑO","INDICE ORDINARIO"
```

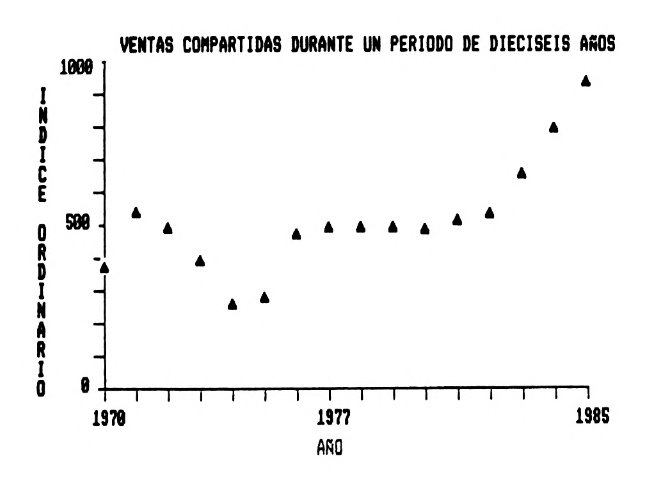


Figura 5,5 Resultado de SUPERG: gráfica de puntos

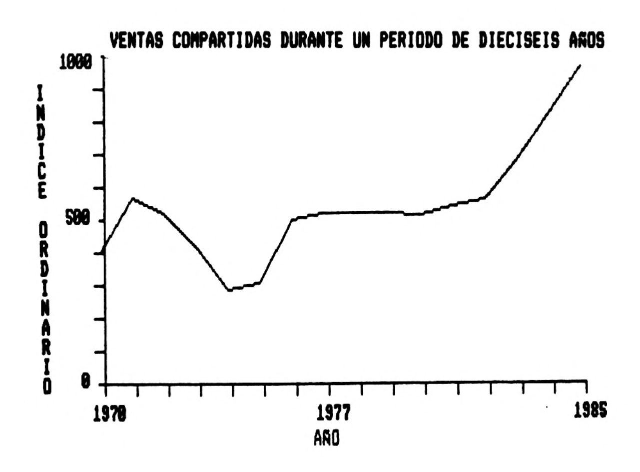


Figura 5,6 Resultado de SUPERG: gráfica de líneas

SUPERG utiliza el modo 2 para dar una apariencia más limpia y clara, pero la mayoría de las gráficas tienen tan buen aspecto en monocromo como en color, por lo que la pérdida del color no es problema alguno en este caso. Puede verse que, una vez introducidos los valores máximos y mínimos de los ejes X e Y, el programa ajusta automáticamente la escala de los datos a las coordenadas de la pantalla, utilizando esa información. La versión anterior de SUPERG representa cada dato como un punto individual. Si en lugar de ello se deseara representar una línea continua (Figura 5.7), hebría que efectuar una pequeña modificación entre las líneas 580-610. Sustituya las líneas 600-610 por

```
600  IF I=1 THEN MOVE 96+(450*(XTRUE/XTOP)),86+(300*(YTRUE/YTOP))
610  DRAW 96+(450*(XTRUE/XTOP)),86+(300*(YTRUE/YTOP))
```

Suele ser necesario representar las fluctuaciones de los datos durante un corto periodo de tiempo. La técnica empleada en este caso es mas o menos la misma que en SUPERG, pero en este caso la etiqueta para el eje del tiempo queda más bonita. Para ello puede emplearse el siguiente programa, llamado TABLA. Si sus dedos están ya hartos de teclear, y si le sirve de consuelo, la mayoría de los programas del resto de este capítulo son sólo variaciones de TABLA, por lo que bastará mezclar (MERGE) las modificaciones con el programa principal.

Programa TABLA

```
10 REM ****PROGRAMA TABLA****
20 REM dibuja una tabla rotulada para un periodo de 12 meses
30 REM por ejemplo, para representar ventas, fluctuaciones de la demanda,
etc
40 REM los datos están al final del programa, pero la entrada interactiva
de los mismos exige una simple modificación
50 DIM xp(12),yp(12)
60 CLS:MODE 2:INK 0,13:INK 1,0
70 REM entrada de rotulos
80     INPUT "TITULO PRINCIPAL (Maximo 80 caracteres)";t$
90     INPUT "TITULO LATERAL (Maximo 20 caracteres)";s$
95     INPUT"SUBTITULO LATERAL (Maximo 20 caracteres)";s1$
100 REM ahora calcula las posiciones de los titulos
110    t1=LEN(t$)
120    t2=LEN(s$)
125    t3=LEN(s1$)
130    xt=40-(t1/2)
140    xs=10-(t2/2)
145    xs1=10-(t3/2)
150 CLS
160    LOCATE xt,2:PRINT t$
170    LOCATE xs,12:PRINT s$
175    LOCATE xs1,13:PRINT s1$
180    GOSUB 450:REM pone en pantalla las leyendas de los meses
190 REM ahora dibuja los ejes
```

```
200  MOVE 145,365:DRAW 145,105
210  DRAW 550,105
220 REM hace las graduaciones del eje Y
230  FOR y=362 TO 112 STEP -25
240  MOVE 142,y:DRAW 147,y
250 NEXT y
260 REM dibujo de la tabla
270 READ mxx
280  xf=131
290  TAG:MOVE 105,360:PRINT mxx;
300  MOVE 105,235:PRINT mxx/2;
310  MOVE 105,112:PRINT 0;
320  FOR i=1 TO 12
330    READ valor
340    valor=((valor/mxx)*250)+112
350    xf=xf+32
360    IF i=1 THEN x2=xf:y2=valor
370    x1=x2:y1=y2
380    x2=xf:y2=valor
390    MOVE x1,y1
400    DRAW x2,y2
420  NEXT i
430 |COPY:REM Este es un comando del programa Tascopy
440 GOTO 440
450 REM leyendas de los meses
460 LOCATE 1,19
470 PRINT TAB(21);"|  |  |  |  |  |  |  |  |  |  |  |"
480 PRINT TAB(21);"E  F  M  A  M  J  J  A  S  O  N  D"
490 PRINT TAB(21);"N  E  A  B  A  U  U  G  E  C  O  I"
500 PRINT TAB(21);"E  B  R  R  Y  N  L  O  P  T  V  C"
510 RETURN
520 DATA 10
530 DATA 1.6,1.8,2.5,2.7,1.1,3.6,4.6,5.9,7.2,8.1,7.1,9.3
```

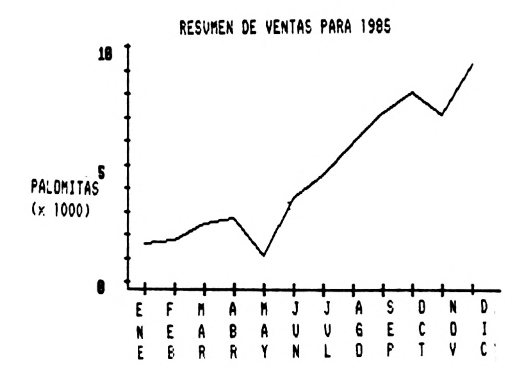


Figura 5.7 Resultado de TABLA

Los datos están contenidos en sentencias DATA, que se leen y representan en las líneas 320-420. Este programa ilustra las variaciones durante un periodo de doce meses, con los meses en el eje X. La leyenda del eje Y se imprime también horizontalmente.

A veces es útil utilizar una gráfica para comparar dos conjuntos de datos. En el caso más sencillo, las dos líneas pueden dibujarse mediante una sencilla modificación de SUPERG o TABLA, dibujando las líneas en distintos colores o con distintos sombreados, o empleando diferentes iconos para representar los datos, en caso de ilustrar puntos discretos. Si la comparación supone algún tipo de expresión competitiva, puede ser de utilidad enfatizar la relación entre las líneas de datos. En la Figura 5.9 se muestra un ejemplo de ello. El periodo durante el cual uno de los productos es el dominante se destaca en un color determinado.

Ventas de dos juegos de ordenador

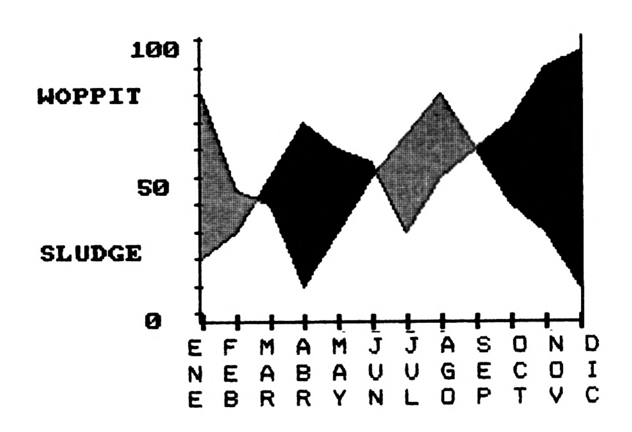


Figura 5,8 Resultado de ENFASIS

Programa ENFASIS

```
10 REM****PROGRAMA ENFASIS****
20 REM version del programa tabla que resalta la comparacion entre dos conjuntos de datos
30 REM Dibuja una gráfica rotulada para un periodo de doce meses
40 REM por ejemplo, para representar ventas, fluctuaciones de demanda,,
50 REM Los datos estánal final del programa, pero la introuccion interactiva requiere solo un sencillo cambio
```

```
70 REM Esta versión rellena las areas situadas entre ambos conjuntos de datos
80 REM con un color diferente, segun cual sea la que supera a la otra
90 REM la orden FILL solo esta disponible en el CPC 664/6128
100 REM se utiliza el modo 1
110 DIM xp(12),yp(12),comp(2,12):REM COMP sirve para guardar pares de puntos de datos
120 CLS:MODE 1:INK 0,13:INK 1,0:INK 2,2:INK 3,20
130 REM entrada de rotulos
140   INPUT "TITULO PRINCIPAL (Max 40 caracteres)";t$
150   INPUT "TITULO LATERAL (Max 20 caracteres)"";s$
160   INPUT"SUBTITULO LATERAL (Max 10 caracteres)";s1$
170 REM calculo de las posiciones de los titulos
180   t1=LEN(t$)
190   t2=LEN(s$)
200   t3=LEN(s1$)
210   xt=20-(t1/2)
220   xs=5-(t2/2)
230   xs1=5-(t3/2)
240   CLS
250   LOCATE xt+1,1:PRINT t$
260   LOCATE xs,6:PRINT s$
270   LOCATE xs1,15:PRINT s1$
280   GOSUB 700:REM colocacione de las leyendas de los meses en pantalla
290 REM ahora dibuja los ejes
300   MOVE 164,365:DRAW 164,105
310   DRAW 520,105
320   MOVE 520,105:DRAW 520,365
330 REM ahora hace las graduaciones del eje Y
340   FOR y=362 TO 112 STEP -25
350   MOVE 161,y:DRAW 167,y
360   NEXT y
370 REM dibuja la primera linea
380 READ mxx
390   xf=135
400   TAG:MOVE 85,360:PRINT mxx;
410   MOVE 90,235:PRINT mxx/2;
420 MOVE 98,112:PRINT 0;
430   FOR i=1 TO 12
440     READ valor
450     valor=((valor/mxx)*250)+112
460     comp(1,i)=valor:REM carga el valor del punto para su posterior comparacion
470     ymin=115
480     xf=xf+32
490     IF i=1 THEN x2=xf:y2=valor
500     x1=x2:y1=y2
510     x2=xf:y2=valor
520     MOVE x1,y1
530     DRAW x2,y2
540   NEXT i
550 REM dibuja la segunda linea
560   xf=135
```

```
570      FOR i=1 TO 12
580      READ valor
590      valor=((valor/mxx)*250)+112
600      comp(2,i)=valor:REM carga el valor del punto para su posterior  comparacion
610      xf=xf+32
620      IF i=1 THEN x2=xf:y2=valor
630      x1=x2:y1=y2
640      x2=xf:y2=valor
650      MOVE x1,y1
660      DRAW x2,y2
670      NEXT i
680 GOSUB 800:REM rellena las areas con color
690 |COPY:END
700 REM leyendas de los meses
710 LOCATE 1,19
720 PRINT TAB(11);"I I I I I I I I I I I I"
730 PRINT TAB(11);"E F M A M J J A S O N D"
740 PRINT TAB(11);"N E A B A U U G E C O I"
750 PRINT TAB(11);"E B R R Y N L O P T V C"
760 RETURN
770 DATA 100
780 DATA 80,45,40,10,30,50,65,80,60,40,30,10
790 DATA 20,30,50,70,60,55,30,50,60,70,90,95
800 REM Ahora rellena las áreas de color con el comando FILL
810      xx=135
820      FOR i=1 TO 12
830        IF comp(1,i)<comp(2,i) THEN col=2
840        IF comp(1,i)>comp(2,i) THEN col=3
850        IF comp(1,i)=comp(2,i) THEN xx=xx+32:GOTO 900
860        yy=(comp(1,i)+comp(2,i))/2
870        xx=xx+32
880        MOVE xx,yy
890        FILL col
900      NEXT i
910 RETURN
```

VENTAS ACUMULADAS DE VEHICULOS NUEVOS Y USADOS - 1984

Figura 5.9 Gráfica acumulativa dibujada con ACUMUL, la mejora acumulativa del programa GRAFICA.

ENFASIS emplea también la orden FILL, por lo que su utilidad será más limitada en el caso del CPC464. Utiliza también el modo 1, produciendo una salida de aspecto bastante más basto.

ENFASIS destaca la diferencia entre dos conjuntos de datos. No obstante, con frecuencia es interesante apreciar el efecto acumulativo de varios conjuntos de datos. La Figura 5.9 nos muestra un ejemplo de ello. La línea inferior es la correspondiente al primer conjunto de datos. La línea superior no es la verdadera línea del segundo conjunto de datos, sino la que representa el total acumulado de ambos conjuntos.

Para utilizar el programa acumulativo ACUMUL, escriba y guarde las siguientes líneas, y combínelas (MERGE) a continuación con el programa TABLA, ya en memoria.

Programa ACUMUL

```
10 REM ****PROGRAMA TABLA****
12 REM ampliacion acumulativa
15 ymin=400
45 REM esta version dibuja una grafica acumulativa para dos conjuntos de
datos
46 REM el conjunto inferior de datos esta sombreado
260 REM ahora dibuja la gráfica superior
270 READ mxx
280 xf=131
290 TAG:MOVE 105,360:PRINT mxx;
300 MOVE 105,235:PRINT mxx/2;
310 MOVE 105,112:PRINT 0;
320 FOR i=1 TO 12
330   READ valor
340   valor=((valor/mxx)*250)+112
345   ymin=115
350   xf=xf+32
360   IF i=1 THEN x2=xf:y2=valor
370   x1=x2:y1=y2
380   x2=xf:y2=valor
390   MOVE x1,y1
400   DRAW x2,y2
420 NEXT i
422 REM ahora dibuja la linea inferior
426   xf=131
428   FOR i=1 TO 12
430     READ valor
432     valor=((valor/mxx)*250)+112
434     xf=xf+32
436     IF i=1 THEN x2=xf:y2=valor
438     x1=x2:y1=y2
439     x2=xf:y2=valor
440     MOVE x1,y1
```

```
442      DRAW x2,y2
444    NEXT i
446    GOSUB 1000:REM Rellena el area superior
448    GOTO 448
520 DATA 10
530 DATA 3.6,5.8,4.5,6.7,3.1,5.6,5.6,6.9,7.9,8.9,6.1,9.3
540 DATA 1.1,1.3,2.1,1.3,0.6,1.2,3.3,3.2,4.3,5.1,3.2,4.1
1000 REM valores de sombreado de la linea superior
1005 inc=4
1010    yval=ymin-2:REM posicion del motivo que constituye el patron de
sombreado
1015    xx=250+inc
1020    FOR yy=yval TO 400 STEP 2
1030    IF TEST(xx,yy)<>0 THEN 1100
1040    PLOT xx,yy
1065    NEXT yy
1100    FOR yy=yval TO 0 STEP -2
1110    IF TEST(xx,yy)<>0 THEN 1200
1120    PLOT xx,yy
1130    NEXT yy
1200    xx=xx+inc
1210    IF xx>515 THEN inc=-inc:xx=250
1220    IF xx<165 THEN RETURN
1230    GOTO 1020
```

Para alivio de los propietarios de un CPC464 (y para permitir utilizar el modo 2), la parte sombreada de la gráfica utiliza una rutina de tramado en el propio programa (la rutina de la línea 1000). Esta rutina va barriendo el eje X, mediante un incremento elegido que se guarda en INC (línea 1005), y dibuja una línea vertical en cada paso desde el eje X (Y=0) hasta el punto en el que la línea vertical corta a la línea inferior de la gráfica.

5.4 Gráficos de barras

Una técnica muy útil para interpretar gráficas de datos consiste en representar los datos como "barras" en lugar de como puntos. Para mejorar la apariencia de las gráficas pueden utilizarse el color y los sombreados. La forma más simple de una gráfica de barras es la que produce el programa BARRAS que se ofrece a continuación. Se trata de una mejora del programa TABLA, y debe mezclarse (MERGE) con el programa TABLA contenido en memoria.

Programa BARRAS

```
10 REM **** AMPLIACION DEL PROGRAMA TABLA QUE DIBUJA BARRAS ****
20 REM Dibuja un diagrama de barras rotulado para un periodo de 12 meses
260 REM ahora dibuja las barras
270 READ mxx
280   xf=131
```

```
290     TAG:MOVE 105,360:PRINT mxx;
300     MOVE 105,235:PRINT mxx/2;
310     MOVE 105,112:PRINT 0;
320     FOR i =1 TO 12
340       READ valor
340       valor=((valor/mxx)*250)+112
350       xf=xf+32
360       REM dibuja el rectangulo para este mes
370       MOVE xf-8,valor:DRAW xf+8,valor
380       DRAW xf+8,112
390       DRAW xf-8,112
400       DRAW xf-8,valor
410       MOVĖ xf,115:fill 1
420     NEXT i
```

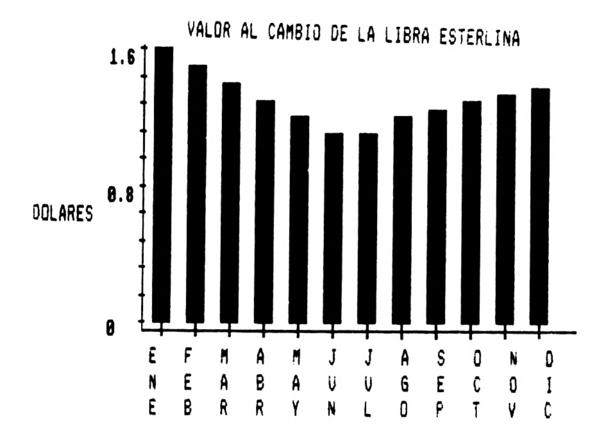


Figura 5.10 Resultado del programa BARRAS

Como puede comprobarse, los cambios son mínimos: en lugar de dibujar una serie de puntos, se genera una serie de rectángulos ("barras"). En este programa las barras se colorean con el comando FILL del CPC664/6128. Los usuarios de un CPC464 pueden dejar vacías las barras, o rellenar cada barra con una trama. He aquí un sencillo programa PATRON que permite rellenar rectángulos con diversos patrones de trama (Figura 5.11).

Programa PATRON

```
10 REM Programa de demostración del tramado de rectángulos
20 CLS
25 LOCATE 12,2:PRINT"EJEMPLOS DE TRAMADO PARA GRAFICAS DE BARRAS"
30 READ xl,xr,yb,yt
35 READ xa,ya,sep
40 IF xl=1 THEN |COPY:END
1020  MOVE xl,yt
1030  DRAW xl,yb
1040  DRAW xr,yb
1050  DRAW xr,yt
1060  DRAW xl,yt
1080 REM Primero rellena la sección superior
1085 IF xa/ya THEN 1165
1090   FOR l=xr TO xl STEP -sep*3
1100     x=l
1110     y=yt
1120     PLOT x,y
1130     x=x-xa
1140     y=y-ya
1150     IF y>=yb AND x>xl THEN 1120
1160   NEXT l
1165 IF ya/xa>2 THEN 1260
1170 REM Ahora rellena la sección inferior
1180   FOR l=yt TO yb STEP -sep*3
1190     y=l
1200     x=xr
1210     PLOT x,y
1220     y=y-ya
1230     x=x-xa
1240     IF y>=yb AND x>xl THEN 1210
1250   NEXT l
1260 GOTO 30
2000 DATA 100,150,300,350,1,1,2
2010 DATA 200,250,300,350,1,1,4
2020 DATA 300,350,300,350,1,1,6
2030 DATA 400,450,300,350,4,4,2
2040 DATA 500,550,300,350,4,4,4
2050 DATA 100,150,200,250,4,4,6
2060 DATA 200,250,200,250,8,8,2
2070 DATA 300,350,200,250,8,8,4
2080 DATA 400,450,200,250,8,8,6
2090 DATA 500,550,200,250,1,8,3
2100 DATA 100,150,100,150,1,8,1
2110 DATA 200,250,100,150,1,8,2
2120 DATA 300,350,100,150,8,1,4
2130 DATA 400,450,100,150,3,5,4
2140 DATA 500,550,100,150,6,2,4
2145 DATA 1,1,1,1,1,1,1
```

EJEMPLOS DE TRAMADO PARA GRAFICOS DE BARRAS

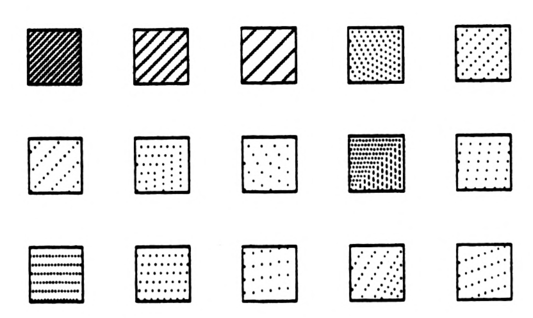


Figura 5.11 **Patrones de tramado producidos con el programa PATRON**

PATRON genera la salida que se ve en la Figura 5.11. Para utilizar este método con un programa de gráficos de barras debe emplearse la sección comprendida entre las líneas 1080 y 1250 como subrutina del programa. Por ejemplo, en BARRAS debe sustituirse la orden FILL 1 por GOSUB 5000, poniendo la sección PATRON a partir de la línea 5000. Antes de llamar a esta subrutina, deben asignarse a XL,XR,YB,YT (X izquierda, X derecha, Y inferior, Y superior) los valores de las cuatro esquinas de la barra. XA,YA y SEP deben ajustarse al patrón de sombreado requerido. La siguiente versión de BARRAS, llamada TRAMA, nos muestra el proceso completo. Una vez más, debe combinarse (MERGE) con el programa TABLA en memoria.

Programa TRAMA

```
405  xl=xf-8:xr=xf+8:yt=valor:yb=115:REM  Carga  de  las  esquinas  para
tramado
410 MOVE xf,115:GOSUB 5000:REM trama
5000 REM Valores de trama
5005 xa=1:ya=1:sep=2:REM Establece los valores de trama
5010 IF xa/ya>2 THEN 5100
5020  FOR l=xr TO xl STEP -sep*3
5030    x=l
5040    y=yt
5050    PLOT x,y
5060    x=x-a
5070    y=y-a
5080    IF y>=yb AND x>xl THEN 5050
5090  NEXT l
```

```
5100 IF ya/xa >2 THEN RETURN
5110 REM Ahora rellena la seccion inferior
5120  FOR l=yt TO yb STEP -sep*3
5130    y=l
5140    x=xr
5150    PLOT x,y
5160    y=y-a
5170    x=x-a
5180    IF y>=yb AND x>xl THEN 5150
5190  NEXT l
5200 RETURN
```

El tramado o el coloreo adquieren una gran importancia cuando se desea utilizar un gráfico de barras para presentar una visualización acumulativa o comparativa de varios conjuntos de datos. La Figura 5.13 nos muestra un método comparativo. De nuevo se emplea el programa básico TABLA, pero, en lugar de modificar BARRAS, deben efectuarse los siguientes cambios:

Programa BARCOMP

```
10 REM **** BAR  -  AMPLIACIONES DEL PROGRAMA TABLA ****
20 REM Dibuja un diagrama de barras con dos conjuntos de datos para
compararlos
25 REM para cada mes se leen a la vez dos puntos de dato
260 REM Ahora dibuja las barras
265 READ mxx
270  xf=129
275  TAG:MOVE 105,360:PRINT mxx;
280  MOVE 105,253:PRINT mxx/2;
285  MOVE 105,112:PRINT 0;
290   FOR i=1 TO 12
295   READ valor
300   valor=((valor/mxx)*250)+112
305   xf=xf+32
310   REM ahora dibuja el primer rectangulo
315   MOVE xf-8,valor:DRAW xf+8,valor
320   DRAW xf+8,112
325   DRAW xf-8,112
330   DRAW xf-8,valor
335   MOVE xf,115:FILL 1
340   REM ahora dibuja el segundo rectangulo para este mes
345   READ valor
350   valor=((valor/mxx)*250)+112
355   xf=xf+6
360   REM desplaza 6 pixels con respecto al primer rectangulo
365   MOVE xf-8,valor:DRAW xf+8,valor
370   DRAW xf+8,112
375   DRAW xf-8,112
380   DRAW xf-8,valor
385   xf=xf-6
```

```
390   NEXT i
400 REM sobreescribe esta linea
420 REM sobreescribe esta linea
520 DATA 100
530 DATA 50,40, 60,20, 70,10, 65,45, 45,75, 10,67
540 DATA 40,75, 50,50, 60,23, 56,43, 45,67, 54,10
```

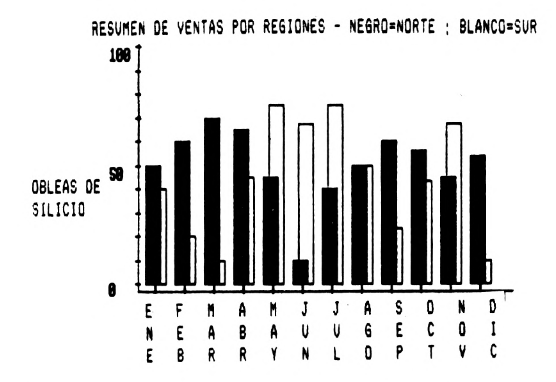


Figura 5.12 **Dos conjuntos de datos de barras dibujados en los mismos ejes mediante la mejora del programa BAR.**

Al igual que antes, este código debe combinarse (MERGE) con el programa TABLA ya en memoria. Puede utilizarse TRAMA con este programa si se desea.

5.5 Gráficos de barras en tres dimensiones.

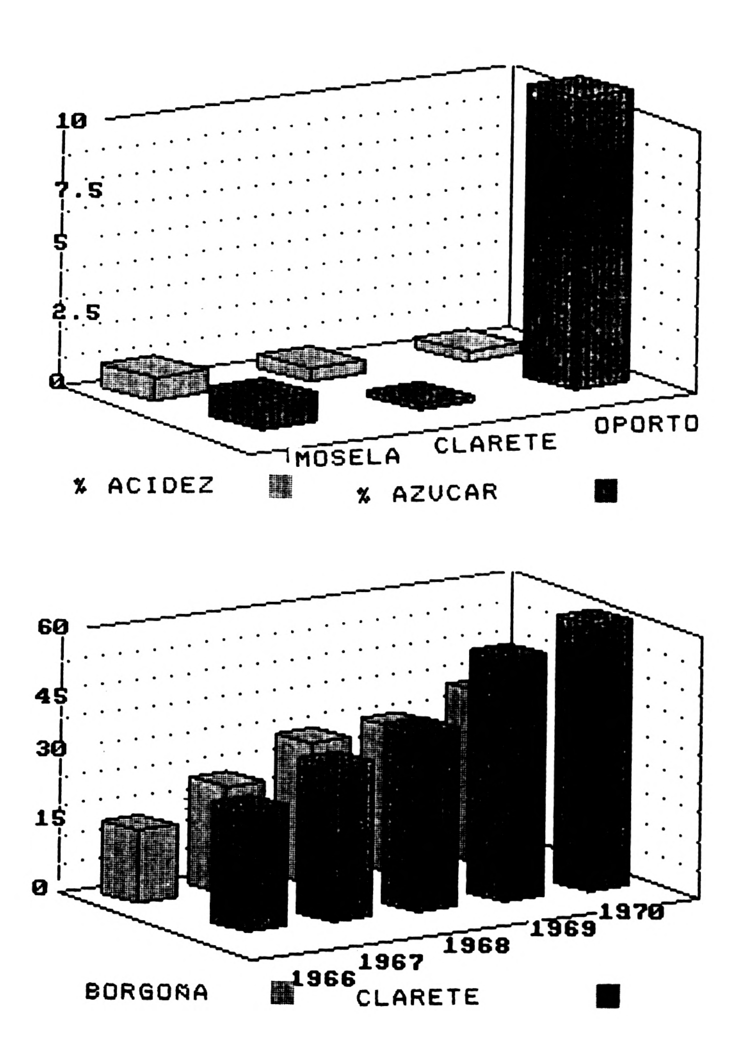
También es posible dibujar un "gráfico de barras tridimensional", para observar el comportamiento de tres variables simultáneamente. El programa HISTO3D que se ofrece a continuación permite crear este tipo de representación. Una salida típica de este programa es la que se muestra en las Figuras 5.13 y 5.14. Este programa no es verdaderamente tridimensional: en realidad produce el efecto de volumen mediante una pequeña trampa que consiste en ampliar los ejes X e Y.

Programa HISTO3D

```
10 REM ****HISTO3D****
20   CLS
30   MODE 1:INK 0,13:INK 1,0:INK 2,9:INK 3,15
40  REM Dibujo del fondo
50   GRAPHICS PEN 1
60   x=460:xl=40:xr=639
70   MOVE x,400
80   DRAW x,160
90   MOVE xl,340
```

```
100  DRAW xl,100
110  MOVE xr,340
120  DRAW xr,100
130  a=1000
140  rw=4
150  c$="E"
160  FOR y=400 TO 160 STEP -(160/6)
170    MOVE x,y
180    MASK 16
190    DRAW xl,y-60
200    IF c$="O" THEN 270
210    LOCATE 2,rw
220    PRINT a;
230    c$="E"
280    MOVE x,y
290    DRAW xr,y-60
300  NEXT y
310  MOVE 40,100
320  DRAW 220,40
330  DRAW 639,100
340  MASK 255
350    LOCATE 17,23:PRINT "1970";
360    LOCATE 26,22:PRINT "1975";
370    LOCATE 35,21:PRINT "1980";
380  h=40:c=0
390  xl=80
400  yl=100
410  xs=240
420  mr=-50/210
430  ml=15/89
440  f=3:o=2
450  h=100
460 REM Dibujo de los bloques
470  GRAPHICS PEN f
480  FOR x=xs TO xs+20
490  MOVE x,20
500  DRAW x,1
510  NEXT x
515  LOCATE 5,25:PRINT"FRANCIA"
516  LOCATE 21,25:PRINT"ALEMANIA"
520     FOR j=1 TO 3
530     READ da:REM captura de datos
540     tp=(240*da)/900
550     GOSUB 680:REM barras de colores
560     GOSUB 980:REM trazado de las barras
570     xl=xl+146
580     yl=yl+20:REM saltos de las barras
590     NEXT j
600  c=c+1
605  LOCATE 1,1
```

```
610  IF c=2 THEN END
620    xl=182
630    yl=75:REM saltos de las barras de la siguiente fila
640    f=2
650    o=3
660    xs=544
670  GOTO 460
680 REM Relleno del recinto
690    bl=yl+tp-ml*xl
700    br=yl-mr*xl
710    FOR x=xl TO xl+h/2
720      y1=ml*x+bl
730      y2=mr*x+br
740      GRAPHICS PEN f
750      MOVE x,y1
760      DRAW x,y2
770      GRAPHICS PEN 1
780      PLOT x,y1
790      PLOT x,y2
800    NEXT x
810  yt=y1
820  yb=y2
830    bl=yb-ml*(xl+h/2)
840    br=yt-mr*(xl+h/2)
850 FOR x=xl+h/2 TO xl+h
860      y1=mr*x+br
870      y2=ml*x+bl
880      GRAPHICS PEN f
890      MOVE x,y1
900      DRAW x,y2
910      GRAPHICS PEN 1
920      PLOT x,y1
930      PLOT x,y2
940    NEXT x
950  yr=y2
960  RETURN
970 REM Rutina de trazado de la barra
980  MOVE xl,yl
990  GRAPHICS PEN 1
1000 DRAW xl,yl+tp
1010 MOVE xl+h/2,yb
1020 DRAW xl+h/2,yb+tp
1030 MOVE xl+h,yr
1040 DRAW xl+h,yr+tp
1050 MOVE xl+h/2,yb+tp
1060 DRAW xl+h,yr+tp
1070 MOVE xl,yl+tp
1080 DRAW xl+h/2,yr+tp-(6*(h/80))
1090 RETURN
1100 DATA 500,500,750,250,500,750
```



Figuras 5.13.5,14 Resultado de HISTO3D. Datos del "Atlas Mundial del Vino" de Hugh Johnson (Mitchell Beazley)

Este es el esquema de HISTO3D:

LINEAS

10- 50	SELECCIONA LOS COLORES, EL MODO Y BORRA LA PANTALLA
60- 340	DIBUJA LA ESTRUCTURA
350- 370	DIBUJA LA ESCALA DEL EJE X
380- 450	DEFINE VARIABLES:
	H=ANCHURA DE LAS BARRAS
	MR,ML=PENDIENTES DE LOS EJES X E Y
	F,O=MODOS DE TINTA PARA LA REPRESENTACION
	XL,YL=PUNTO INFERIOR IZQUIERDO DE LA BARRA
	XS=POSICION DE LOS BLOQUES DE CODIGO DE COLOR
460- 510	DIBUJO DE LOS BLOQUES DE CODIGO DE COLOR
515- 516	DIBUJO DE LAS LEYENDAS DE LOS CODIGOS DE COLOR
520- 670	BUCLE PRINCIPAL
600	INCREMENTO DEL CONTADOR DE FILAS
605	COLOCACION DEL CURSOR DE TEXTO EN LA POSICION INICIAL
610	COMPRUEBA SI SE HAN HECHO TODAS LAS FILAS
620- 660	PREPARACION DE VARIABLES PARA LA SIGUIENTE FILA
670	FIN DEL BUCLE PRINCIPAL
680- 960	RUTINA PARA COLOREAR UNA BARRA
	BL,BR=VALORES DE LA Y PARA LOS EXTREMOS IZQUIERDOS DE LA BARRA
970-1090	RUTINA PARA TRAZAR LA BARRA
1100	DATOS (PARA CADA FILA, DE IZQUIERDA A DERECHA)

Una vez ajustadas las alturas de las barras, las rutinas de las líneas 680 y 970 trazan y rellenan las de cada punto. Como las barras traseras se dibujan primero, se produce un efecto de "superficie oculta" (ver Capítulo 8), ya que las barras delanteras se dibujan sobre parte de las posteriores, dibujadas anteriormente.

El comando MASK del CPC664 se utiliza en las líneas 180 y 340 para dibujar la rejilla de las coordenadas. Si dispone de un CPC464, puede utilizar el programa TRAZOS (Capítulo 2) para dibujar las líneas de puntos. Si se desea, también pueden dibujarse rectas continuas. Por supuesto, el valor de MASK puede cambiarse si es necesario.

Existen muchas otras variantes de los gráficos comerciales que no hemos considerado en este capítulo. Es posible, por ejemplo, visualizar al mismo tiempo una tarta y una gráfica, o dos gráficos de barras a la vez, uno extendido por encima del eje Y y otro por debajo de él. Dejamos a su imaginación la programación de estas y otras variaciones: los métodos se basarán en los principios que ya hemos examinado.

Capítulo 6

Un programa de Diseño Asistido por Ordenador

6.1 Consideraciones acerca del diseño

La mayor parte del material que hemos considerado hasta ahora constituye un campo de trabajo de gran utilidad para la manipulación de imágenes bidimensionales en la pantalla del ordenador. Aparte del área comercial, no hemos visto aún ninguna aplicación "seria" de los gráficos en dos dimensiones; aunque ya comentábamos las limitaciones que presenta el BASIC para numerosas aplicaciones gráficas, es hora de poner manos a la obra y ver qué es lo que podemos crear.

Los gráficos por ordenador se emplean extensivamente en el mundo profesional para trabajos de diseño asistido por ordenador (CAD), en los que pueden diseñarse estructuras complejas configurando figuras o diagramas a partir de un cierto número de subcomponentes. Estos componentes pueden manipularse a su vez en la pantalla para modificar o actualizar la estructura a construir. Ya nos hemos encontrado con las transformaciones geométricas que funcionan como las materias primas necesarias para realizar una versión simplificada de un paquete de diseño sobre un microordenador; el problema sigue siendo cómo agrupar los componentes en un programa coherente. Este problema se debe sobre todo a la complejidad de las estructuras de datos que guardan los elementos empleados en la figura.

El programa, al que naturalmente llamaremos DISEÑO, es lo bastante flexible como para necesitar escasas modificaciones para diseñar planos de casas, trazados de decoración, mapas del tiempo y, en fin, todo lo que implique expresar en dos dimensiones configuraciones espaciales. Las oportunidades de aplicarlo a nuestros problemas particulares irán quedando claras a medida que el programa vaya desarrollándose.

6.2 ¿Qué es lo que queremos?

Decidamos en primer lugar el conjunto de elementos gráficos simples que deben incluirse en un paquete de diseño de este tipo. En primer lugar, es imprescindible una concepción general del objeto a diseñar. Para nuestros propósitos actuales, supondremos que cada objeto es una región del

espacio limitada por una serie de puntos: un polígono o una serie de polígonos, hablando en términos geométricos.
Podemos repasar ahora nuestro arsenal de técnicas, para saber lo que está a nuestra disposición (y lo que nos proponemos hacer con el programa DISEÑO). Nuestras principales aliadas serán las rutinas del programa SKETCH, y con la técnica de "barrido de líneas" dibujaremos el trazado general y las demarcaciones más importantes del sistema. Los elementos más pequeños que se colocarán en distintos puntos del trazado se definirán también de forma separada mediante las rutinas de SKETCH, pero combinando las transformaciones bidimensionales simplificaremos el acceso a los elementos y su dibujo. Las coordenadas de cada elemento se guardarán en nuestras queridas matrices coordenadas X e Y, y la matriz V cumplirá la misma misión que siempre, es decir, coordinar la conexión de los pares coordenados mediante líneas.

La principal sofisticación de las estructuras de datos empleadas en DISEÑO es la matriz S, la cual, si recordamos el Capítulo 3, sirve para ir controlando los diversos segmentos que pueden usarse. Volviendo a la sección 2 del Capítulo segundo, recordemos que S estaba definida como una matriz de 2 x n, donde n es el número de segmentos. DISEÑO utiliza el primer segmento para representar el trazado de la estructura sobre la cual va a efectuarse el diseño, en este caso los límites y las principales demarcaciones. Los demás elementos se emplean para los diversos componentes que deben situarse en el área de diseño.

Además de llevar el control sobre las configuraciones y tamaños de los distintos segmentos a manipular en la pantalla, es necesario recordar las posiciones de los segmentos. Recordemos también del Capítulo 3 que las cuatro sillas de la figura de demostración de los segmentos eran en realidad el mismo segmento repetido cuatro veces. Ninguna de las estructuras de datos que hemos visto hasta ahora proporciona información alguna capaz de reconstruir múltiples apariciones de un mismo segmento, ni siquiera para visualizar los segmentos en distintos puntos de la pantalla. Con esta finalidad, el programa DISEÑO utiliza una nueva matriz, dimensionada como RD(3,i). En este caso i representa el número total de elementos de que consta la figura (cada aparición de un segmento se cuenta como un elemento más), y para cada segmento se graban las coordenadas X e Y del punto central del elemento, junto con el número de segmento del elemento. De este modo, para una figura que contenga cinco elementos elegidos de tres segmentos, la matriz RD tendrá una apariencia similar a esta:

X	**Y**	**segmento número**
140	100	3
100	80	1
120	10	1
200	60	2
200	130	2

Como veremos al ir construyendo el programa DISEÑO, las estructuras de datos de que ahora disponemos (X,Y,W,S y RD) nos permitirán construir y manipular figuras complejas.

La tabla RD permite situar elementos en pantalla mediante un solo par de coordenadas. ¿Cómo es esto posible? A diferencia de los segmentos de datos considerados en el programa SKETCH original, los datos de un segmento en el programa DISEÑO se trasladan al origen en el mismo momento de su creación, utilizando una técnica que es básicamente igual a la del Capítulo 4: se calcula el punto central del segmento, y todas las coordenadas dentro del segmento se decrementan por los valores X e Y del punto central. De este modo colocamos, efectivamente, el punto central en la posición 0,0, y todos los demás puntos a su alrededor. Para centrar en el origen el segmento trasladado, de una forma elegante, tomaremos como centro el punto cuyas coordenadas sean los valores medios de las Xs e Ys. extremas de la figura. Observe que no tiene por qué existir físicamente ese punto central de la figura; es una mera referencia.

Cuando debe representarse en pantalla un segmento, puede ser dibujado en torno a cualquier punto X,Y, con solo sumar a las coordenadas almacenadas X,Y del segmento el valor +X, +Y.

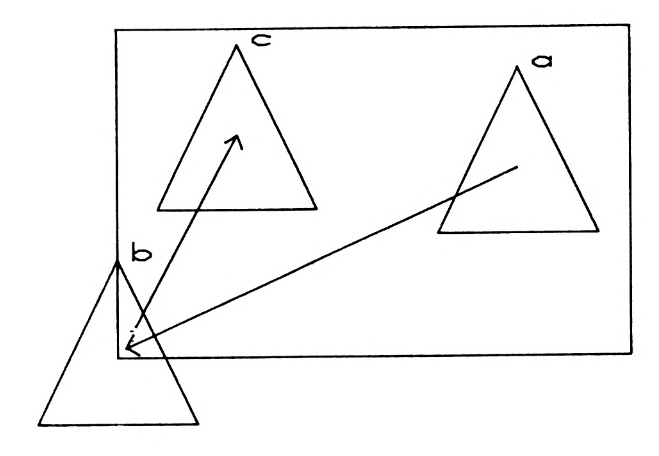


Figura 6.1 Creación y traslación de segmentos. Un segmento puede dibujarse en cualquier punto de la pantalla y trasladarse después al origen (-X,-Y). El segmento puede entonces dibujarse de nuevo en cualquier punto X1 Y1 de la pantalla sumando X1,Y1 a todos los valores de las coordenadas del segmento en el origen.

Una vez definidas las estructuras de datos que emplearemos en DISEÑO, debemos considerar la estructura de control del programa. Debido a su complejidad, no cabe utilizar un pequeño conjunto de instrucciones memorizadas para manejar el programa, como sucedía con SKETCH. Podemos necesitar cargar o guardar ficheros o partes de ficheros, definir segmentos o dibujar trazados. Para poder controlar el flujo entre estos diversos estados, emplearemos un sistema de menús. Seguramente ya estará Vd. familiarizado con los programas guiados por menús de diverso género. En DISEÑO utilizaremos dos sencillos menús. El primero de ellos permite seleccionar las principales funciones disponibles en el programa. Un segundo menú ofrece varias modalidades de carga de ficheros. Estos son los dos menús tal y como los presenta el programa:

DISEÑO - MENU PRINCIPAL

DIBUJAR EL TRAZADO DE LA IMAGEN	-1
DEFINIR ELEMENTOS	-2
GRABAR LA FIGURA	-3
GRABAR SOLO ELEMENTOS	-4
CARGAR FIGURA	-5
DIBUJAR LOS ELEMENTOS DEL TRAZADO	-6
BORRAR ELEMENTOS ANTES DE DIBUJARLOS	-7
IMPRIMIR LA FIGURA	-8

Figura 6,2 El menú principal de DISEÑO

DISEÑO - MENU DE CARGA

CARGAR SOLO EL TRAZADO	-1
CARGAR SOLO LOS ELEMENTOS	-2
CARGAR TRAZADO + ELEMENTOS	-3
CARGAR TODA LA FIGURA	-4

Figura 6,3 El menu de carga de DISEÑO

La mayoría de las opciones disponibles a través de estos dos menús están contenidas en subrutinas o en grupos de subrutinas. Este método modular

nos permitirá construir el diagrama de flujo del programa DISEÑO. Este es:

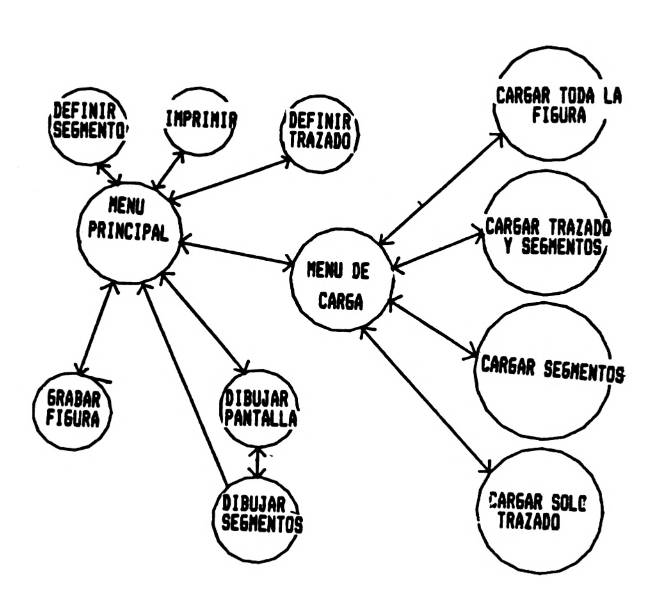


Figura 6.4 Sencillo diagrama de flujo del programa DISEÑO. Todas las opciones están controladas por los dos menús.

Como puede verse en este diagrama, las principales funciones de DISEÑO son:

(1) Visualización de menú
(2) Definición de segmentos
(3) Definición del trazado
(4) Dibujo de segmentos en el trazado
(5) Grabación de datos
(6) Carga de datos
(7) Impresión de la figura

Para llevar a cabo estas funciones es preciso tener en cuenta una serie de detalles técnicos:

(1) Actualización de contadores y variables para cada estructura de datos.
(2) Movimiento y presentación del cursor, y actualización de la posición actual del cursor en la pantalla.
(3) Traslación y cambio de escala de segmentos.
(4) Construcción de figuras a partir de los datos disponibles.
(5) Facilidades de control no guiadas por menú.

Más adelante en este mismo capítulo, inmediatamente antes del listado del programa, ofrecemos una lista de los contadores y otras variables empleadas en DISEÑO. Las rutinas de manejo del cursor son iguales que las de SKETCH. Ya hemos descrito en esta sección las traslaciones de segmentos, pero el cambio de escala de los mismos requiere una explicación más detallada.

Como el tamaño final en pantalla de algunos segmentos puede ser demasiado pequeño, convendría poder definir (dibujar) cada segmento empleando unas dimensiones "cómodas", para luego comprimirlo a las dimensiones adecuadas. En el programa DISEÑO esto se hace dibujando cada segmento en una celdilla ampliada equivalente a las pequeñas celdillas que aparecen en el margen derecho de la pantalla principal de diseño, como puede verse en las Figuras 6.5 y 6.6 que aparecen a continuación.

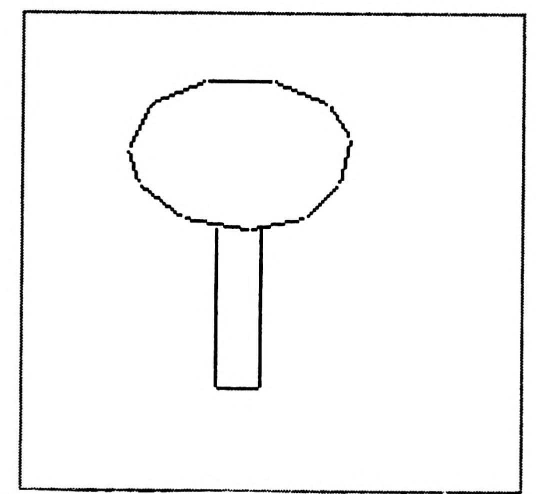


Figura 6.5 Dibujo de un segmento a escala ampliada.

Los métodos de construcción de la imagen manejan las matrices RD y S para acceder a los índices de las líneas que se guardan en la matriz W, y en último extremo a las coordenadas X e Y que se almacenan en las matrices X e Y. Si debe dibujarse el elemento número SP, las coordenadas X e Y del punto alrededor del cual debe dibujarse el segmento, junto con el tipo de segmento (SN), vienen dadas por la matriz RD, de modo que

```
RD(1,SP)=X,RD(2,SP)=Y,RD(3,SP)=SN
```

Una vez obtenida esta información, es posible acceder a las matrices S,W,X e Y para dibujar el segmento SN alrededor del punto X,Y, empleando el siguiente código del programa DISEÑO:

```
FOR I=S(1,SN) TO S(2,SN)
L1=W(1,I):L2=W(2,I)
MOVE XP(L1)+X,YP(L1)+Y
DRAW XP(L2)+X,YP(L2)+Y
NEXT I
```

Observe que las matrices XP e YP contienen los datos coordenados en el origen. Sumando X a los elementos XP e Y a los elementos YP, el segmento se desplaza a la posición correcta.

Obviamente, no es posible acceder a todas las funciones por medio de menús, puesto que es preciso dar algunos pasos de control durante la creación de la imagen - por ejemplo, las líneas deben ser comenzadas y terminadas. Al igual que sucedía con SKETCH, es necesario definir los puntos de comienzo y de término de las nuevas líneas, y acceder a segmentos concretos durante el proceso de creación sería un proceso tedioso si el ordenador tuviese que estar permanentemente conmutando entre la pantalla en alta resolución y los menus en baja resolución. Como en SKETCH, es necesario emplear algo el teclado (para concluir un segmento, o para señalar un "salto" dentro de un segmento. Además, se utiliza el botón de disparo del *joystick*. A continuación echaremos un vistazo a la facilidad más interesante en términos de gráficos por ordenador - el empleo de parte de la propia pantalla de alta resolución para funciones de control. Esta posibilidad sólo se activa en modo diseño, en el que los segmentos disponibles se muestran en el borde derecho de la pantalla. La Figura 6.6 nos ilustra el empleo del programa de diseño. Puede ser conveniente acudir a esta figura para entender mejor los conceptos que se exponen en el texto.

Si el cursor se desplaza a una de las celdillas que contienen los segmentos, puede colocarse una copia de ese segmento en cualquier lugar de la pantalla con sólo pulsar el botón de disparo. El segmento dibujado puede especificarse de nuevo llevando el cursor a otra celdilla. Podemos ver también las palabras "SAL" y "RELL". La primera de ellas es autoexplicativa, mientras que la segunda hará que toda la figura comience de nuevo.

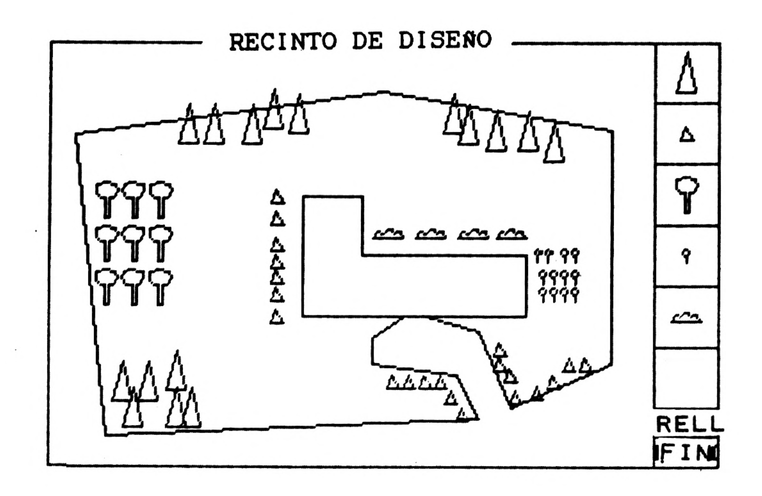


Figura 6.6 Volcado de una pantalla DISEÑO que muestra el trazado de un jardín.

Este tipo de representación visual con fines de control está ganando popularidad en áreas como los gráficos comerciales, en los que se emplea con frecuencia el famoso "ratón", dispositivo relacionado con el *joystick*, para desplazar el cursor por la pantalla y seleccionar las opciones disponibles dentro de un programa o de una serie de ellos. Se trata, en efecto, de una técnica especialmente sencilla de programar. Un simple test dentro del programa comprueba si los valores de la X y de la Y de la posición del cursor están comprendidos dentro de ciertos límites y, de ser así, salta a otra parte del programa.

6.3 El programa DISEÑO

Antes del listado de DISEÑO, he aquí una relación de las variables que se emplean dentro del programa

INDICADORES (BANDERAS O FLAGS)/CONTADORES EN GENERAL

FL	bandera que indica que una línea está en marcha
NPTS	contador del número de puntos que contiene un segmento
NA	contador de puntos totales
LB	contador de líneas totales
SE	bandera indicadora de fin de segmento
FI	bandera para corte de línea

PF   bandera indicadora del tipo de segmento (trazado/elemento)
LI   contador de líneas del segmento
S   contador del número de elementos dibujados
CS   tamaño del cursor
SS   tamaño de los pasos del cursor
N$ nombre del fichero de entrada
H$ nombre del fichero de salida
K$ cadena de caracteres introducidos por teclado
I,J   contadores generales de incremento
PW   anchura del segmento en pixels
WI   factor de reducción de la anchura del segmento

VARIABLES COORDENADAS

X   coordenada X del cursor
Y   coordenada Y del cursor
X1   valores coordenados temporales de la X
Y1   valores coordenados temporales de la Y
X1   coordenada X al comienzo de la línea
Y1   coordenada Y al comienzo de la línea
XF   coordenada X al final de la línea
XH,XL   valores máximos y mínimos de la X para un segmento
YH,YL   valores máximos y mínimos de la Y para un segmento
XC,YC   punto central del segmento
XX,YY   valores coordenados temporales de la X y la Y

MATRICES

XP(i)
YP(i)   matrices que guardan los valores coordenados X,Y

W(2,j) matriz que almacena los índices de comienzo y final de cada línea
S(2,k) matriz que contiene los índices de la primera y la última línea de cada segmento.

RD(3,l)   matriz que contiene las coordenadas X e Y de cada aparición de un segmento, junto con el tipo de segmento.

Programa DISEÑO

```
5 REM ****PROGRAMA DISEÑO****
10  cs=2:ss=5
15 MODE 1
30  INK 0,13:INK 1,1:INK 2,3
45  CLS
```

```
50  REM Definicion del tamaño y pasos del cursor
60  fl=0:npts=1:na=1
70  lb=0:REM contador de lineas
80  se=0:REM indicador de fin de segmento
90  sl=0:REM contador de segmentos
91  sp=0:REM contador de elementos de la figura
92  fi=0:REM indicador de corte de linea
93  li=0:REM contador de lineas del segmento
94  jy=1:REM indicador de comienzo/final de linea
95  pf=1:REM indicador del tipo de segmento
100 DIM xp(500),yp(500),ln(2,500),s(3,10),rd(3,100):REM Dimensionamiento de
matrices
110 GOTO 1690:REM vuelta al menu principal
120 REM colocacion del cursor en la posicion central
125 IF pf=2 THEN GOSUB 2900
130 x=320:y=200
140   GOSUB 180:REM rutina de dibujo del cursor
150   GOSUB 230:REM rutina de movimiento del cursor
160   GOSUB 340:REM rutina de barrido de la linea
170 GOTO 140
180 REM rutina de dibujo del cursor
190 x1=x-cs:y1=y-cs:x2=x+cs:y2=y+cs
200   MOVE x1,y
205   DRAW x2,y,1,1
210   MOVE x,y1
215   DRAW x,y2,1,1
220 RETURN
230 REM rutina de movimiento del cursor
240 y3=y:x3=x
250   IF JOY(0)=0 THEN 310
260   IF JOY(0)=1 THEN y=y+ss:GOTO 310
270   IF JOY(0)=2 THEN y=y-ss:GOTO 310
280   IF JOY(0)=4 THEN x=x-ss:GOTO 310
290   IF JOY(0)=8 THEN x=x+ss:GOTO 310
310     MOVE x3,y2
320     DRAW x3,y1,1,1
325     MOVE x1,y3
326     DRAW x2,y3,1,1
330 RETURN
340 REM rutina de dibujo y barrido de la linea
350 a$=INKEY$
355 IF a$="" AND JOY(0)<>16 THEN IF fl=0 THEN RETURN
370 IF JOY(0)=16 AND jy=1 THEN jy=2:LOCATE 2,2:PRINT "S":GOSUB 3000:GOTO
430
380 IF JOY(0)=16 AND jy=2 THEN jy=1:LOCATE 2,2:PRINT "F":GOSUB 3000:GOTO
460
390 IF a$="B" THEN jy=1 THEN jy=1:GOTO 450:REM corte de linea
400 IF a$="E" THEN se=1:jy=1:GOTO 460:REM fin de la figura
420 GOTO 650:REM dibujo/borrado normal de la linesa
430 xi=x:yi=y:REM coordenadas de comienzo
```

```
440 fl=1
450 fi=1:REM indicador de corte de linea
460 xf=x:yf=y:REM asignacion de punto
480   MOVE xi,yi
485   DRAW xf,yf
490 npts=npts+1:na=na+1:li=li+1:lb=lb+1:REM incrementa contadores
500 xp(na)=xf:yp(na)=yf:REM  asignacion de punto
510 xp(na-1)=xi:yp(na-1)=yi:REM asignacion de puntos
560 ln(1,lb)=na-1:REM asignacion de indices de linea
570 ln(2,lb)=na
580 IF fi=1 THEN na=na+1:fi=0:REM incremento si el indicador de corte
esta activado
590 IF se=1 THEN s1=s1+1:s(1,s1)=npts-li:s(2,s1)=npts-1:s(3,s1)=0:GOTO 690
630 fl=0:RETURN
640 fl=0
650 REM realizacion del dibujo/borrado de la linea
660    MOVE x,y
665    DRAW xi,yi,1,1
670    MOVE x,y
675    DRAW xi,yi,1,1
680 RETURN
690 REM continuar
710 FOR i=s(1,s1) TO s(2,s1)
730    MOVE xp(ln(1,i)),yp(ln(1,i))
735    DRAW xp(ln(2,i)),yp(ln(2,i)),1,0
740 NEXT i
750 k$=INKEY$:IF k$="" THEN 750
780 li=0:fl=0:se=0:na=na+1:REM si, entonces actualiza los contadores
790 IF pf=1 THEN 1690:REM trazado, entonces volver al menu principal
792 GOSUB 1000:REM comprimir el elemento de la figura
794 GOTO 1690:REM vuelta al menu principal
796 GOTO 120
800 REM ahora crea el fichero que contiene los datos
810 CLS
820 INPUT"nombre del fichero de salida";n$
830 OPENOUT n$
840 WRITE#9,na
850 FOR i=1 TO na
860 WRITE#9,xp(i)
870 WRITE#9,yp(i)
875 NEXT i
880 WRITE#9,lb
890 FOR i=1 TO lb
900   WRITE#9,ln(i,i)
910   WRITE#9,ln(2,i)
915 NEXT i
920  WRITE#9,s1
930 FOR i=1 TO s1
940   WRITE#9,s(1,i)
950   WRITE#9,s(2,i)
```

```
965 NEXT i
970    PRINT#9,sp
980    FOR i=1 TO sp
982      PRINT#9,rd(1,i),rd(2,i),rd(3,i)
984    NEXT i
986 CLOSEOUT
1000 REM comprime el elemento de la figura
1030 REM imprime "tamaño del elemento:"
1040 REM INPUT "Anchura en pixels?";PW
1060 REM define punteros maximos y minimos
1070     xh=0:xl=640:yh=0:yl=400
1080     l1=ln(1,s(1,s1)):l2=ln(2,s(2,s1))
1085       FOR i=l1 TO l2
1090         IF xp(i)<xh THEN xl=xp(i)
1100         IF xp(i)>xh THEN xh=xp(i)
1110         IF yp(i)<yl THEN yl=yp(i)
1120         IF yp(i)>yh THEN yh=yp(i)
1130       NEXT i
1140 REM ajusta la anchura
1145 wi=0.16
1150 REM calcula el punto central para trasladarlo al origen
1160   xc=((xh+xl)/2)*wi:yc=((yh+yl)/2)*wi
1170 REM ahora reduce el tamaño del objeto y lo lleva al origen
1180   FOR i=l1 TO l2
1185     xp(i)=(xp(i)*wi)-xc
1186     yp(i)=(yp(i)*wi)-yc
1187   NEXT i
1190 RETURN
1200 REM coloca en pantalla los segmentos
1210     sm=0:x=320:y=200
1220     GOSUB 180:REM rutina de dibujo del cursor
1230     GOSUB 230:REM rutina de movimiento del cursor
1235       IF x>574 THEN GOSUB 1900:REM escoger segmento
1240       IF JOY(0)=16 THEN GOSUB 1260:REM dibujar segmento
1245       IF x<4 THEN GOTO 1690:REM regreso al menu principal
1250     GOTO 1220:REM regreso del bucle
1260 REM dibujo del segmento
1265  sp=sp+1:rd(1,sp)=x:rd(2,sp)=y:rd(3,sp)=sm
1270  FOR i=s(1,sm) TO s(2,sm)
1280    l1=ln(1,i):l2=ln(2,i)
1285        MOVE xp(l1)+x,yp(l1)+y
1290        DRAW xp(l2)+x,yp(l2)+y,2,0
1300  NEXT i
1310  RETURN
1330  REM rutina de entrada de fichero
1340  INPUT"nombre del fichero de entrada";h$
1360    OPENIN h$
1370    INPUT#9,npts
1380    FOR i=1 TO npts
1390      INPUT#9,xp(i),yp(i)
```

```
1395     NEXT i
1400     INPUT#9,li
1420     FOR i=1 TO li
1425       INPUT#9,ln(1,i),ln(2,i)
1427     NEXT i
1430     INPUT#9,s1
1440     FOR i=1 TO s1
1450       INPUT#9,s(1,i),s(2,i)
1460     NEXT i
1465     INPUT#9,sp
1470     FOR i1 TO sp
1475       INPUT#9,rd(1,i),rd(2,i),rd(3,i)
1480     NEXT i
1485 CLOSEIN
1490 PRINT"fichero",h$,"cargado correctamente"
1500 RETURN
1560 REM rutina de dibujo de los límites de la hoja de diseño
1570 MOVE 6,394
1575   DRAW 146,394
1580   MOVE 440,394
1590   DRAW 634,394
1600   DRAW 634,6
1610   DRAW 6,6
1620   DRAW 6,394
1630   MOVE 574,394
1635   DRAW 574,6
1640   FOR i=60 TO 340 STEP 56
1645     MOVE 574,i
1650     DRAW 634,i
1655   NEXT i
1660   MOVE 574,34
1665   DRAW 634,34
1670   LOCATE 13,1:PRINT"RECINTO DE DISEÑO"
1675   IF pf=1 THEN LOCATE 1,1:PRINT"TRAZADO"
1676 RETURN
1680 REM MENU PRINCIPAL
1690 CLS:pf=2
1700 PRINT
1710 PRINT"               DISEÑO - MENU PRINCIPAL"
1720 PRINT"     DIBUJAR EL TRAZADO DE LA IMAGEN       - 1"
1740 PRINT"     DEFINIR ELEMENTOS                     - 2"
1750 PRINT"     GRABAR LA FIGURA                      - 3"
1755 PRINT"     GRABAR SOLO LOS ELEMENTOS             - 4"
1760 PRINT"     CARGAR FIGURA                         - 5"
1770 PRINT"     DIBUJAR LOS ELEMENTOS DEL TRAZADO     - 6"
1780 PRINT"     BORRAR ELEMENTOS ANTES DE DIBUJARLOS  - 7"
1790 PRINT"     IMPRIMIR LA FIGURA                    - 8"
1792 PRINT"     SALIR DEL PROGRAMA                    - 9"
1794 k$=INKEY$: IF k$="" THEN 1794
1796     IF k$="1" THEN pf=1 :CLS:GOSUB 1570:GOTO 120:REM definir trazado
```

```
1800     IF k$="2" THEN pf=2 :CLS:GOTO 120:REM definir elemento
1810     IF k$="3" THEN GOTO 810:REM grabar todo
1815     IF k$="4" THEN GOSUB 2700:GOTO 1680:REM grabar solo los
elementos
1820     IF k$="5" THEN sp=0:GOSUB 2300:GOTO 1690:REM cargar todo
1830     IF k$="6" THEN CLS:GOSUB 2000:GOSUB 1570:GOSUB 2100:GOTO 1210
1840     IF k$="7" THEN CLS:sp=0:s1=1:GOSUB 2000:GOSUB 1570:GOSUB
2100:GOTO 1210
1850     IF k$="8" THEN |COPY:GOTO 1680:REM volcado de pantalla con
Tascopy
1860     IF k$="9" THEN PRINT"PROGRAMA ABANDONADO":END
1870 GOTO 1794
1900 REM rutina de eleccion de segmento para su dibujo
1910     IF y>340 THEN sm=2:RETURN
1920     IF y>284 THEN sm=3:RETURN
1930     IF y>228 THEN sm=4:RETURN
1940     IF y>172 THEN sm=5:RETURN
1950     IF y>116 THEN sm=6:RETURN
1960     IF y>60 THEN |copy:sm=7:RETURN
1965     IF y>32 THEN GOSUB 2200:x=x-30:RETURN:REM regenerar la figura
1970     IF y<34 THEN GOTO 1690
1990     IF JOY(0)<>16 THEN RETURN
2000 REM rutina de dibujo de segmento
2010     ya=424
2020     IF s1=1 THEN 2085
2027      FOR i=2 TO s1
2030        ya=ya-56
2040        FOR j=s(1,i) TO s(2,i)
2060         MOVE xp(ln(1,j))+604,yp(ln(1,j))+ya
2065         DRAW xp(ln(2,j))+604,yp(ln(2,j))+ya,2,0
2070        NEXT j
2080       NEXT i
2082 TAG
2085 MOVE 574,54:PRINT"RELL "
2090 MOVE 574,26:PRINT"SAL  "
2092 TAGOFF
2095 RETURN
2100 REM regeneracion del trazado del diseño
2110  FOR i=s(1,1) TO s(2,1)
2120   l1=ln(1,i):l2=ln(2,i)
2130   MOVE xp(l1),yp(l1)
2135   DRAW xp(l2),yp(l2),1,0
2140  NEXT I
2150  RETURN
2200  REM rutina de regeneracion de la figura
2215   IF sp=0 THEN GOSUB 2100:RETURN
2220    FOR I=1 TO sp
2230      xx=rd(1,i):yy=rd(2,i)
2235      FOR j=s(1,rd(3,i)) TO s(2,rd(3,i))
2240        l1=ln(1,j):l2=ln(2,j)
```

```
2260            MOVE xp(l1)+xx,yp(l1)+yy
2265            DRAW xp(l2)+xx,yp(l2)+yy,2,0
2267        NEXT j
2270      NEXT i
2275 GOSUB 2100:REM regeneracion del trazado
2280 RETURN
2300 REM menu de carga
2310 CLS
2320 pf=2
2330 PRINT""
2340 PRINT"                DISEÑO  -  MENU DE CARGA"
2350 PRINT""
2360 PRINT"        CARGAR SOLO EL TRAZADO              - 1"
2370 PRINT"        CARGAR SOLO LOS ELEMENTOS           - 2
2380 PRINT"        CARGAR TRAZADO + ELEMENTOS          - 3"
2390 PRINT"        CARGAR TODA LA FIGURA               - 4"
2400 k$=INKEY$:IF k$="" THEN 2400
2410   IF k$="1" THEN GOSUB 1330:npts=s(2,1)+1:li=0:s1=1:na=npts+1
2415   IF k$="1" THEN lb=npts-1:sp=0:RETURN
2420   IF k$="2" THEN GOSUB 2500:sp=0:RETURN
2425   IF k$="3" THEN GOSUB 1330:sp=0:RETURN
2430   IF k$="4" THEN GOSUB 1330:RETURN
2500 REM Carga solo los elementos
2510 INPUT"NOMBRE DE FICHERO PARA LOS ELEMENTOS";h$
2520 OPENIN h$
2530 npts=s(2,1)+1:li=0:s1=1:na=npts+1
2540 INPUT#9,nw
2550   FOR i=npts+1 TO npts+nw
2560     INPUT#9,xp(i),yp(i)
2570   NEXT i
2575 npts=npts+nw
2580 INPUT#9,lw
2585   FOR i=lb+1 TO lb+lw
2590     INPUT# 9,ln(1,i),ln(2,i)
2595   NEXT i
2600 lb=lb+lw:li=lb
2605 INPUT 9,s(1,i),s(2,i)
2610 s1=s1+1
2620   FOR i=2 TO s1
2630     INPUT#9,s(1,i),s(2,i)
2640   NEXT i
2650 PRINT"FICHERO",h$,"CARGADO CORRECTAMENTE"
2655 CLOSEIN
2660 CLS
2670 RETURN
2700 REM crea un fichero que contiene solo elementos de datos
2710 CLS
2720  INPUT"NOMBRE DE FICHERO PARA LOS ELEMENTOS";h$
2730  OPENOUT h$
2740  PRINT#9,(na-(ln(2,s(2,1))))-1
```

```
2750       FOR i=(ln(2,s(2,1)))+1 TO na-1
2760         PRINT#9,xp(i)
2770         PRINT#9,yp(i)
2780       NEXT i
2790     PRINT#9,lb-s(2,1)
2800     FOR i=s(2,1)+1 TO lb
2810       PRINT#9,ln(1,i)
2820       PRINT#9,ln(2,i)
2825     NEXT i
2830       PRINT#9,s1-1
2840         FOR i=2 TO s1
2850           PRINT#9,s(1,i)
2860           PRINT#9,s(2,i)
2870         NEXT i
2880       CLOSEOUT
2890   RETURN
2900   REM trazado para el recinto de creacion de segmento
2910   MOVE 140,368
2915   DRAW 500,368,2,0
2920   DRAW 500,32,2,0
2925   DRAW 140,32,2,0
2930   DRAW 140,368,2,0
2940   LOCATE 2,24:PRINT "DIBUJA UN ELEMENTO EN EL RECINTO"
2950   RETURN
3000   FOR i=1 TO 1000:NEXT i:RETURN
```

DISEÑO se emplea de la siguiente forma. Al ejecutar el programa, aparece el menu principal, que permite al usuario cargar o guardar ficheros, definir los segmentos o el trazado o imprimir el área de diseño. Las opciones de definición comienzan en modo "SKETCH"; puede crearse un segmento con el *joystick*, el botón de disparo y las teclas S y E, como con SKETCH. La opción más importante es el propio diseño de la pantalla, que aparecerá como más adelante veremos, con un determinado trazado y los segmentos dibujados a la derecha de la pantalla. El cursor se va moviendo, para tomar y dibujar segmentos individuales, hasta que la figura quede completa. SAL vuelve al menú principal.

El menú de carga permite cargar del disco o la cinta un trazado, un conjunto de segmentos o incluso una figura completa.

El programa DISEÑO queda estructurado como sigue:

```
LINEAS 10-  20 DEFINICION DE LOS TAMAÑOS DEL CURSOR Y LOS PASOS
  30-       45 DEFINICION DE TINTAS Y COLORES Y BORRADO DE LA PANTALLA
  60-      100 INICIALIZACION DE LOS CONTADORES Y LAS BANDERAS
 110           SALTO A LA RUTINA QUE DIBUJA EL MENU PRINCIPAL
 125-          ESCRIBIR "SEGMENTO" SI SE ESTA CREANDO UN SEGMENTO
 130           COLOCACION DEL CURSOR EN EL MEDIO DE LA PANTALLA
```

140-	170	SECCION DE CONTROL DE LA PANTALLA PRINCIPAL
180-	220	RUTINA DE DIBUJO DEL CURSOR
230-	330	RUTINA DE MOVIMIENTO DEL CURSOR
340-	680	SECCION DE DIBUJO Y BARRIDO DE LINEAS
730-	790	COMPROBACION DE FIN DE LINEA O SEGMENTO
810-	990	CREACION DE FICHERO DE DATOS
1000-	1190	DISMINUCION DEL TAMAÑO DEL SEGMENTO
1200-	1310	COLOCACION DE SEGMENTOS EN LA PANTALLA
1320-	1550	RUTINA DE ENTRADA DE FICHERO
1560-	1670	DIBUJO DE LOS LIMITES DE LA PANTALLA
1680-	1785	PANTALLA DE VISUALIZACION DEL MENU PRINCIPAL
1790-	1850	LECTURA DE TECLA Y SALTO
1900-	1910	SELECCION DE SEGMENTO PARA SU DIBUJO
2000-	2095	DIBUJO DE SEGMENTOS DEL MARGEN DERECHO
2100-	2160	DIBUJO DE TODO EL TRAZADO DEL JARDIN
2200-	2300	RUTINA DE REGENERACION DE LA FIGURA
2310-	2390	VISUALIZACION DEL MENU DE CARGA
2400-	2430	LECTURA DE TECLA PARA CARGAR OPCION Y SALTAR A OTRO PUNTO DEL PROGRAMA
2500-	2650	ENTRADA DE FICHERO: SOLO DE SEGMENTOS
2700-	2880	CREACION DE FICHERO DE DATOS (SOLO ELEMENTOS)
3000-	3005	BUCLE DE RETARDO

Estas descripciones son más bien escasas, por lo que en las siguientes notas se dan algunos detalles más sobre las secciones más complejas:

Líneas 20-40

El tamaño del cursor y la magnitud de los saltos pueden modificarse si se desea, aunque, por supuesto, cuanto mayor sea el paso, menos precisa será la imagen final. Un cursor grande impresiona bastante, pero recuerde que la inversión de pixels producirá un fastidioso e innecesario "parpadeo" a lo largo de las líneas del cursor si la figura tiene detalles muy finos.

Líneas 140-790

Esta gran sección es casi idéntica a la del programa SKETCH, y realiza las operaciones de dibujo más importantes.

Líneas 730-790

Es preciso destacar algunas diferencias con respecto a SKETCH. Si el segmento definido es el trazado, se transfiere inmediatamente el control al menú principal. En caso contrario, el segmento se encoge antes de acceder al menú principal.

Líneas 810-990

Esta es la principal rutina de creación de ficheros de datos. Toda la información contenida en las estructuras de datos es almacenada; el trazado, los segmentos y los datos de posición de los segmentos se guardan en el disco.

Líneas 1010-1190

Esta rutina comprime un elemento de la figura, dibujado a gran escala, al tamaño de la "celdilla".

Líneas 1200-1310

La colocación de segmentos en la pantalla requiere tres pasos. En primer lugar, se emplean las rutinas de movimiento y colocación del cursor para permitir seleccionar y situar los segmentos. En segundo lugar, el segmento a dibujar se toma de las opciones que se muestran a la derecha de la pantalla. Para manejar esto se llama a la rutina comprendida entre las líneas 1900-1990. El segmento escogido se dibuja entonces alrededor de la posición actual del cursor (líneas 1260-1310). La matriz RD se incrementa, y las coordenadas X,Y de la posición actual del cursor, junto con el número del segmento, se colocan en esta matriz.

Líneas 1320-1550

Esta es la rutina principal de entrada de ficheros de datos. Lee los datos de X,Y,W,S y RD.

Líneas 1560-1670

Esta rutina dibuja las fronteras de la pantalla y coloca el título en su sitio. Si se está dibujando el trazado, aparece en pantalla la palabra "trazado"

Líneas 1900-1990

El número del segmento actual a dibujar se actualiza si la posición X del cursor es > 287. La coordenada Y del cursor determina el número del segmento. Esta rutina se llama desde la rutina de dibujo de segmento (líneas 1200-1310).

Líneas 2000-2095

Esta rutina dibuja, de arriba abajo, los segmentos disponibles, en la parte derecha de la pantalla, dejándolos preparados para ser seleccionados por la subrutina anterior. Todos los segmentos se dibujan con el mismo desplazamiento en X, y el desplazamiento en Y se incrementa para cada segmento sucesivo.

Líneas 2100-2160

Dibuja el trazado, dejándolo preparado para colocar en él los elementos.

Líneas 2200-2300

Esta rutina dibuja todas las apariciones de todos los elementos en la pantalla. Se accede a ella escogiendo la opción "RELL" en la pantalla de dibujo. Se emplea si se ha salido de la pantalla de dibujo para definir nuevos segmentos, por ejemplo, y se desea acceder de nuevo a ella.

Líneas 2500-2650

La rutina de entrada asume que el trazado ha quedado ya definido, y carga una serie de segmentos en las matrices X,Y,W y S. Observe que esta rutina no rellena RD, porque los datos de los segmentos pueden utilizarse con cualquier número de trazados, mientras que la información de RD es específica de un solo trazado. Observe además que ese trazado ya debería existir. Esto es necesario para evitar el problema de la cantidad de espacio que es necesario dejar al comienzo de las diversas matrices que configuran el trazado.

Líneas 2700-2880

Esta rutina de creación de ficheros no guarda el trazado, y el fichero creado se emplea en conjunción con la rutina anterior.

6.4 Algunas aplicaciones del programa DISEÑO

DISEÑO tiene aplicación en diversas áreas diferentes. La Figura 6.6 nos mostraba una de ellas. Podemos crear también, por ejemplo, nuestros propios mapas del tiempo, dibujando un mapa como trazado y diseñando símbolos para colocarlos en el lugar adecuado (tal vez, para el mapa de la figura, el símbolo de sol no será necesario nunca). También podríamos diseñar jardines, hacer proyectos de decoración...

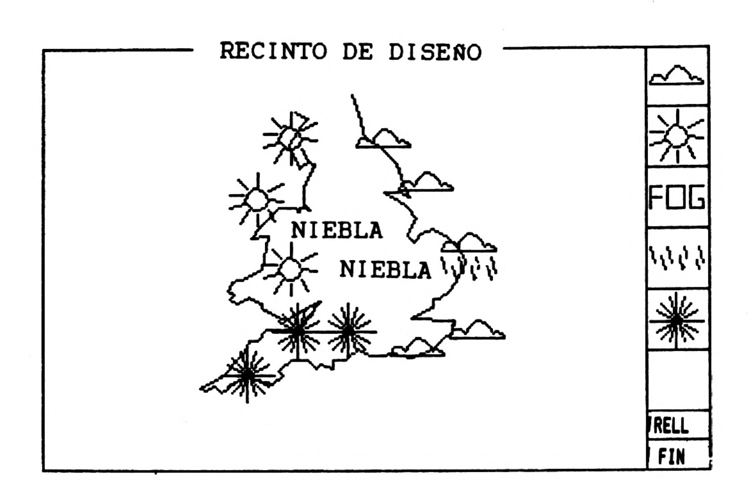


Figura 6.7 Utilización del programa DISEÑO para dibujar un mapa del tiempo.

Capítulo 7

Trabajando en tres dimensiones

7.1 Datos y proyecciones en dos dimensiones

Las técnicas para dibujar líneas que unan puntos de un plano bidimensional son bastante fáciles de dominar, pero los problemas aparecen cuando nos enfrentamos con la tercera dimensión. Estos problemas no son específicos del CPC 6128, CPC 664 ó CPC 464, sino en general de todos los ordenadores, aunque los paquetes de *software* gráfico disponibles para miniordenadores y grandes ordenadores (y algunos micros de elevado coste) pueden disponer de ciertas facilidades para manejar datos tridimensionales, especialmente en lo que respecta a la perspectiva y las proyecciones, cuestiones que consideraremos más adelante en este capítulo. Los temas que trataremos se refieren a los gráficos en "estructura de alambre", ya que sólo consideraremos puntos y líneas, y no superficies como tales. La representación de objetos sólidos será el tema del Capítulo 8.

Para poder representar datos tridimensionales necesitamos, en primer lugar, extender el sistema de coordenadas rectangulares presentado en el capítulo 4. Recordemos que en dos dimensiones tenemos unos ejes X e Y que se extienden a derecha e izquierda y arriba y abajo de la página, respectivamente. En tres dimensiones debemos representar también la profundidad, lo cual se consigue empleando un tercer eje llamado Z. Este eje suele dibujarse en dirección perpendicular a la página. Si colocamos en esa posición un lápiz sobre la página, tendremos una representación del eje Z, como puede verse aquí.

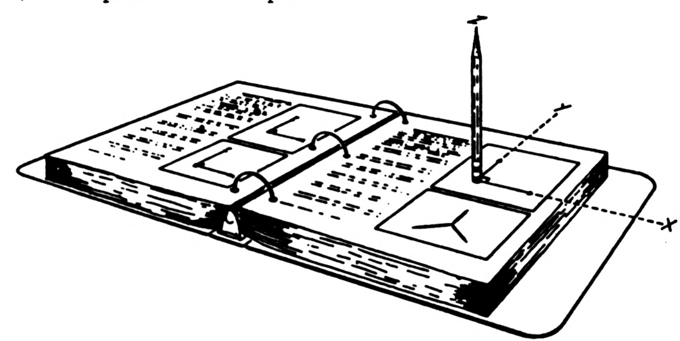


Figura 7.1 Representación del eje Z

Si el observador los viera desde una posición ligeramente desplazada del eje Z, los ejes tendrían este aspecto

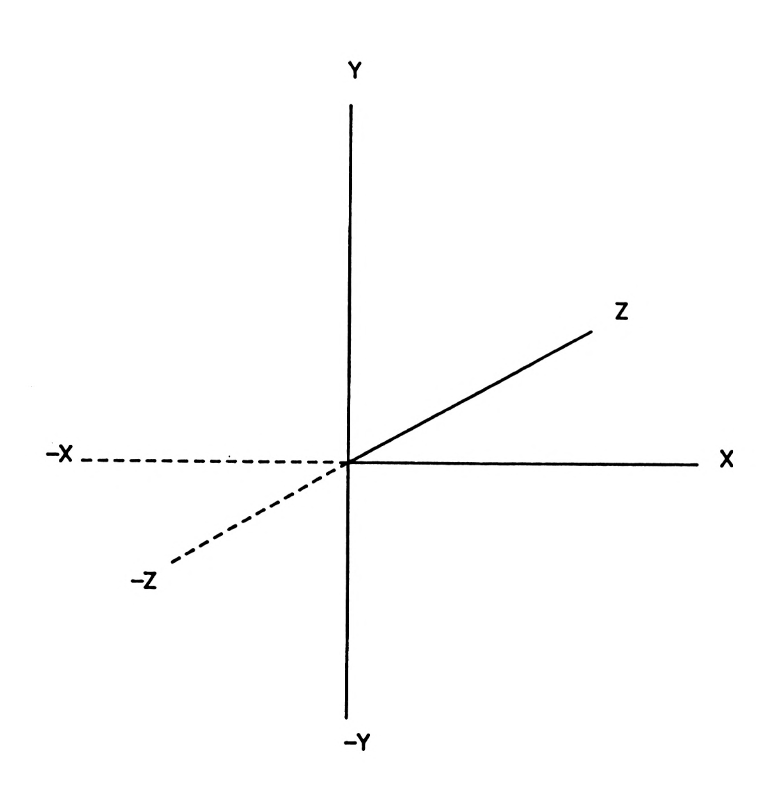


Figura 7.2 Los tres ejes en el espacio. El eje Z negativo se proyecta más allá del plano de la pantalla, hacia el observador.

El dato Z se trata de forma idéntica a los datos X e Y, es decir, tiene su propia matriz unidimensional para almacenar los datos de la coordenada Z de cada punto. Como ejercicio para "pensar tridimensionalmente", consideremos un cubo, quizá el más sencillo de los objetos tridimensionales. Para mayor simplicidad, supongamos también que las coordenadas del cubo en el espacio son mayores que las del origen en todas las dimensiones. El cubo quedará por tanto definido por una serie de puntos, cuyas coordenadas podrían ser, por ejemplo, las siguientes:

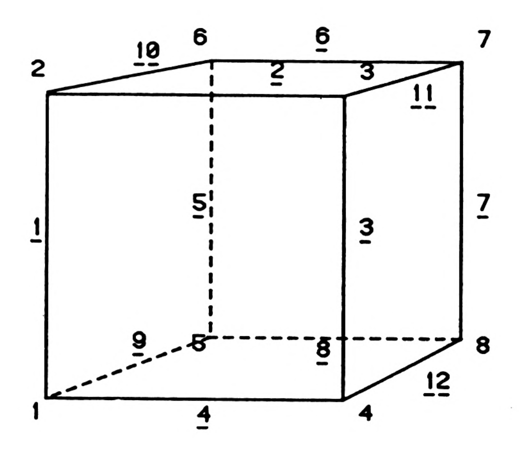


Figura 7,3 Datos para puntos y líneas para el cubo utilizado como ejemplo de 3 dimensiones en este capítulo, Los números de las líneas son los que están subrayados,

pto. no.	*X*	*Y*	*Z*
1	50	50	50
2	50	100	50
3	100	100	50
4	100	50	50
5	50	50	100
6	50	100	100
7	100	100	100
8	100	50	100

Podemos definir también las líneas que deben dibujarse entre los puntos, de manera idéntica a como hicimos en el caso bidimensional, con lo cual

i	*V(1,i)*	*V(2,i)*
1	1	2
2	2	3
3	3	4
4	4	1
5	5	6
6	6	7
7	7	8
8	8	5
9	1	5
10	2	6
11	2	6
12	4	8

Observe que en este caso hay más líneas que puntos

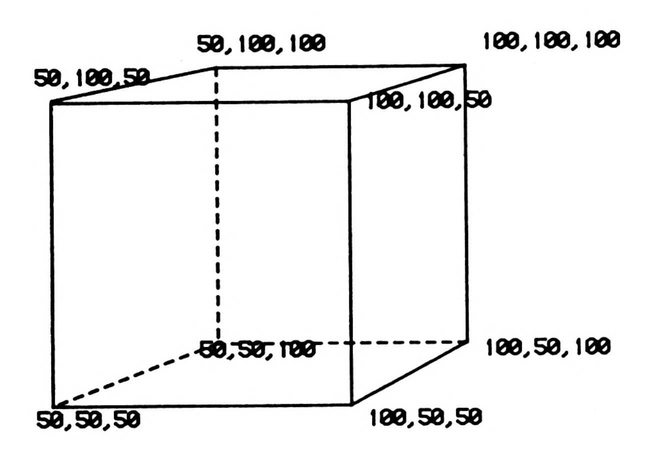


Figura 7.4 Coordenadas X,Y,Z para el cubo de la Figura 7.3.

La matriz W se dimensiona igual en tres dimensiones que en dos, y se emplea también de la misma manera. Por otro lado, los datos X Y Z deben transformarse para eliminar el eje Z antes de dibujar, ya que no hay forma de representar coordenadas X Y Z en una pantalla X Y. No podemos emplear las coordenadas X Y Z para dibujar de la misma forma que empleábamos los datos X Y en el caso bidimensional. Para dibujar, emplearemos las matrices XP e YP que ya introducíamos en la sección 3.2 del Capítulo 3. La cuestión que hemos de resolver es cómo pasar de tres a dos dimensiones.

7.2 Métodos de proyección

Las técnicas que permiten pasar de las coordenadas tridimensionales a una trama coordenada bidimensional se llaman proyecciones. Veamos los dos diagramas que aparecen a continuación. El objeto a proyectar está contenido en un espacio tridimensional llamado *volumen de visión* (análogo a la ventana de dos dimensiones que encontrábamos en el Capítulo 4).

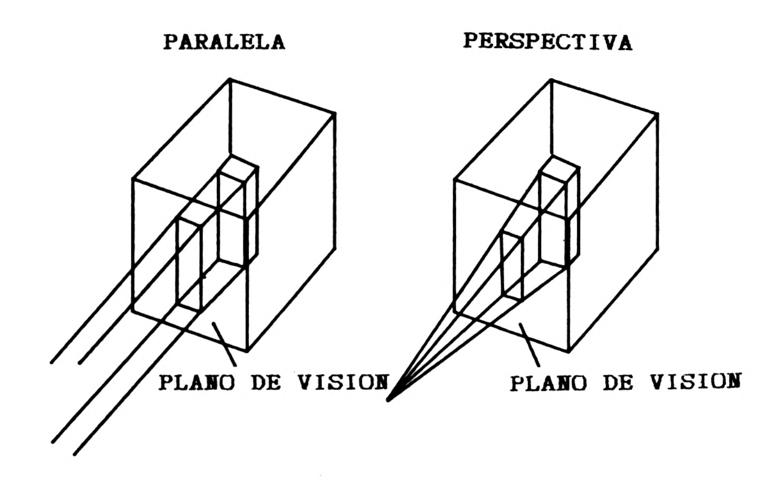


Figura 7.5 Vistas en paralelo y en perspectiva de un bloque rectangular contenido dentro del volumen de visión, el equivalente tridimensional de las ventanas del mundo de dos dimensiones. El plano de visión es el cuadro bidimensional sobre el cual se proyecta el bloque; como puede verse, el resultado es diferente para las proyecciones paralela y en perspectiva.

Hay dos formas distintas de proyectar este volumen en dos dimensiones, dependiendo del eje de proyección. Si la proyección se realiza hacia un punto, tenemos la proyección en *perspectiva*. Si la proyección de todos los puntos del volumen de visión es paralela, entonces (lógicamente) la proyección se llama paralela. Podemos visualizar esta idea simplemente como la diferencia en la posición de un "observador". Si el observador se encuentra a una cierta distancia d del volumen de visión, siendo d < infinito, contemplará una visión en perspectiva, puesto que los rayos de luz procedentes de todos los puntos del objeto convergerán en sus ojos. Si consideramos que el observador está en el infinito - lo que en términos físicos viene a significar una distancia suficientemente grande -, los rayos de luz nunca convergerán. Los raíles de la figura muestran la diferencia entre las visiones en paralelo y en perspectiva.

En general, las proyecciones en perspectiva son más fieles a la realidad, pero bastante más enrevesadas de programar. Las proyecciones en paralelo son, por el contrario, menos "realistas", pero más fáciles de realizar. La técnica a emplear depende, en gran medida, del proyecto de que se trate. Ejecutando los programas de este capítulo podrá comparar los efectos de cada una de ellas.

Volviendo a las proyecciones del volumen de visión que mostrábamos antes, podemos ver un elemento que nos queda por explicar. Este elemento es el plano de proyección, en el que reside la clave de nuestro problema de

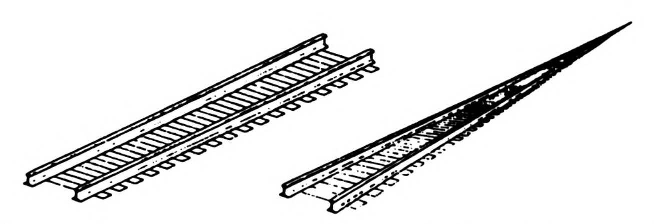


Figura 7.6 Un ejemplo muy ilustrativo de las proyecciones en paralelo y en perspectiva; las vías del tren.

proyección. El plano de proyección se define como el plano en el que se "cortan" las líneas de proyección. Si el plano es paralelo al eje Z, las coordenadas Z se pierden en el plano, ya que todos los puntos del mismo tienen el mismo valor Z. Comparando las perspectivas en paralelo y en perspectiva, vemos que las representaciones en dos dimensiones del plano de proyección son diferentes en cada caso.

Así, la imagen que se presenta al ojo es diferente según cuál sea el tipo de proyección. Antes de echar un vistazo a los distintos métodos de proyección, hemos de advertir que la imagen resultante de la proyección en dos dimensiones dependerá de la posición relativa del "ojo" con respecto al origen. Para nuestros propósitos, supondremos que el ojo mira directamente al eje Z, de forma idéntica a la orientación de los ojos del lector con respecto a la página de este libro. Esto no supone limitación alguna, ya que podemos hacer rotar cualquier objeto de tres dimensiones para observarlo desde cualquier posición. Todo esto quedará más claro más adelante, cuando expliquemos las transformaciones en tres dimensiones.

7.3 Cómo introducir datos tridimensionales

Recordemos las utilidades descritas en el Capítulo 3 para crear ficheros de datos en dos dimensiones (FICHERO2D, SKETCH). Hemos visto ya que un conjunto de datos en tres dimensiones puede crearse con sólo añadir un conjunto adicional de coordenadas Z; así, FICHERO2D se convierte, en la nueva versión, en FICHERO3D.

Programa FICHERO3D

```
10 REM****PROGRAMA FICHERO3D****
20 REM Programa que almacena los datos de las coordenadas a dibujar
25 CLS
30 INPUT"NOMBRE DEL FICHERO";h$
40 OPENOUT h$
50 INPUT"NUMERO DE PUNTOS";npts
55 WRITE #9,npts
60 PRINT "INTRODUZCA LAS TERNAS X,Y,Z"
70 FOR i=1 TO npts
```

```
80 INPUT"X=";x:INPUT"Y=";y:INPUT "Z=";z
90 WRITE #9,x
100 WRITE #9,y
105 WRITE #9,z
110 NEXT i
120 INPUT"NUMERO DE LINEAS";li
130 WRITE #9,li
140 PRINT"Escriba los números de los puntos de union:"
150 FOR i=1 TO li
160 INPUT "Empieza en el punto numero";sn:INPUT"y acaba en el punto
numero";fi
170 WRITE #9,sn
180 WRITE #9,fi
190 NEXT i
200 CLOSEOUT
210 END
```

SKETCH es más enrevesado de adaptar. No se debe, por supuesto, a nada relacionado con la complejidad de establecer un conjunto de puntos en tres dimensiones, sino a la dificultad de "pasar" de la proyección en dos dimensiones de la pantalla a los datos tridimensionales. Podemos hacerlo de dos formas. La primera y más peliaguda consiste en ir definiendo uno a uno los puntos de la pantalla especificando sus coordenadas X Y y Z. Una posible forma de hacerlo podría consistir en programar el *joystick* para desplazar el cursor por la pantalla en los planos X e Y, como en SKETCH. Sin embargo, en lugar de crear una imagen en dos dimensiones, podrían emplearse dos teclas más para gobernar el movimiento del cursor hacia arriba y hacia abajo de la pantalla con el fin de añadir la dimensión Z. Sólo cuando el valor de Z sea el correcto la línea se almacenará como en SKETCH. El quid de la cuestión está, por supuesto, en calibrar Z. El valor por defecto de Z debe definirse como 0, pero Z puede variarse desde, pongamos, -300 hasta +300. Un sistema como este debería utilizar algún tipo de ventana en la parte inferior de la pantalla para mantenernos informados del valor actual de Z. Sería preciso programar algún tipo de indicador de Z en la pantalla de alta resolución. También habría que considerar el problema de la proyección; la solución más sencilla sería la proyección paralela. Debería ser bastante fácil mejorar un programa como éste para que dibuje en perspectiva, pero no sería excesivamente útil si no se es un artista.

Tras leer el párrafo anterior, tal vez esté Vd. pensando, no sin motivo, que este autor ha contraído ese síndrome, tan común entre los autores de textos informáticos, que consiste en describir gran número de programas de jugosa apariencia sin proporcionar el código de ninguno. No se preocupe. La única razón de que no se encuentre Vd. con un programa como los citados anteriormente es que tengo auténticas dudas sobre su utilidad.

En lugar de ello, propongo un programa más elegante, aunque mucho más restrictivo, que nos permitirá añadir a nuestros datos en dos dimensiones alguna información acerca de la profundidad. Este programa, llamado

SKETCH3D, permite definir el trazado de un objeto de la misma forma que SKETCH, pero, una vez dibujado el objeto, se añade el dato Z a los datos de las coordenadas X,Y. Seguidamente, los datos que representan una segunda imagen del trazado se calculan con las mismas coordenadas X,Y, pero con la coordenada Z ajustada a un valor diferente del de la primera imagen del objeto. Además de calcular de esta forma las coordenadas X,Y,Z, se expande la matriz W para que contenga los datos relativos a las líneas que unen el primero y el segundo conjunto de puntos. Los datos disponibles en este momento permiten reconstruir dos objetos planos en el espacio de tres dimensiones, pero esto no es suficiente, ya que lo que en realidad deseamos representar es un solo objeto tridimensional. La matriz W es la clave de este objeto, ya que todo lo que necesitamos es un conjunto de líneas que unan los puntos correspondientes de los dos planos. El diagrama que se ofrece a continuación nos muestra de forma gráfica los pasos necesarios para preparar el conjunto final de datos tridimensionales. Es posible añadir también datos adicionales al conjunto de datos con el fin de emplearlos en combinación con un programa de "líneas ocultas".

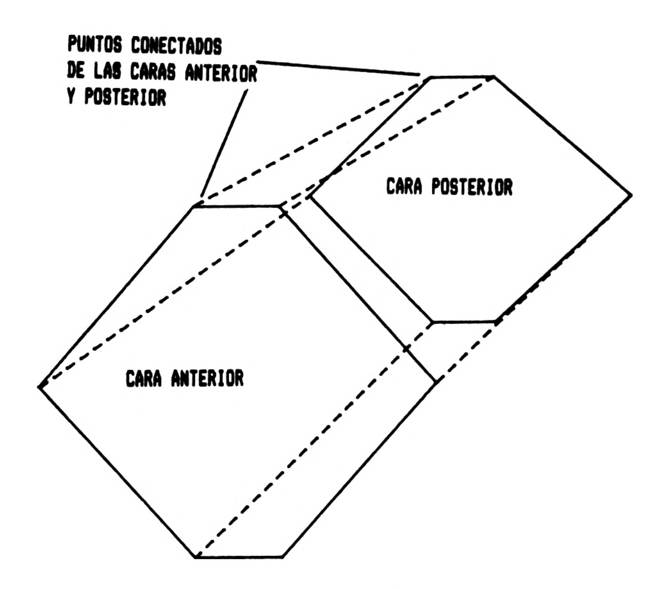


Figura 7.7 Producción de un conjunto de datos de 3 dimensiones a partir de datos bidimensionales. La primera cara se define como una serie de puntos y líneas de dos dimensiones. A continuación se añade el dato Z constante a cada par de coordenadas X,Y. Los datos X,Y se duplican a continuación para producir la cara posterior (con valores de Z iguales a una cierta constante, pero mayores que los de la cara frontal). Por último, se añade un conjunto de líneas que unen los puntos correspondientes de ambas caras.

SKETCH3D es muy agradable de utilizar, ya que evita calcular largas cadenas de datos de tres dimensiones. Está limitado, por supuesto, a aquellos objetos para los que sea válida esta técnica de representación, pero es especialmente útil para adquirir práctica en "percepción espacial". Para convertir a SKETCH en SKETCH3D, deben mezclarse (MERGE) las siguientes modificaciones con SKETCH, que ha de estar previamente en memoria.

Programa SKETCH3D

```
10 REM****PROGRAMA SKETCH3D****
110 s(1,2)=0
800 REM Ahora crea un fichero que contiene los datos
810 OPENOUT n$
815 GOSUB 1200:REM Toma los valores de traslacion
817 PRINT"la cuantia del desplazamiento en X es",xh-xl
818 PRINT"la cuantia del desplazamiento en Y es",yh-yl
819 INPUT"introduzca la cuantia del desplazamiento en Z";zz
820   PRINT #9,na*2
830   FOR  i=1 TO na
840     PRINT #9,xp(i)+xtrans
850     PRINT #9,yp(i)+ytrans
860     PRINT #9,-(zz/2)
870   NEXT i
875 REM sobreescribe esta linea
880   FOR i=1 TO na
890     PRINT #9,xp(i)+xtrans
900     PRINT #9,yp(i)+ytrans
910     PRINT #9,(zz/2)
915 REM sobreescribe esta linea
920   NEXT i
930 IF s(1,2)=0 THEN s(1,2)=lb
940    PRINT #9,(2*lb)+s(1,2)-1
950    FOR i=1 TO lb
960      PRINT #9,ln(1,i),ln(2,i)
965 REM sobreescribir esta linea
970    NEXT i
980    FOR i=1 TO lb
990      PRINT #9,ln(1,i)+na
1000     PRINT #9,ln(2,i)+na
1010   NEXT i
1020   FOR i=1 TO s(1,2)
1030     PRINT #9,i
1040     PRINT #9,i+na
1050   NEXT i
1060   PRINT #9,s1
1070     FOR i=1 TO s1
1080       PRINT #9,s(1,i)
1090       PRINT #9,s(2,i)
1100     NEXT i
```

```
1110  CLOSEOUT
1120  END
1200 REM Rutina para trasladar los valores X,Y hacia el origen
1205 xl=640:yl=400:xh=0:yh=0
1210     FOR i= 1 TO na
1220        IF xp(i)<xl THEN xl=xp(i)
1230        IF xp(i)>xh THEN xh=xp(i)
1240        IF yp(i)<yl THEN yl=yp(i)
1250        IF yp(i)>yh THEN yh=yp(i)
1260     NEXT i
1270  xtrans=-((xh+xl)/2)
1280  ytrans=-((yh+yl)/2)
1290  RETURN
```

Como puede comprobarse examinando un poco el código anterior, la mayoría de los cambios a realizar en SKETCH afectan a la sección de creación de ficheros.

Línea 820

El número de puntos de datos en SKETCH3D es 2*NA, o dos veces el número de puntos dibujados en la figura de dos dimensiones.

Línea 860

Los valores de la coordenada Z para todos los puntos de la cara frontal del objeto en tres dimensiones están ajustados arbitrariamente a -(ZZ/2), y pueden modificarse si es necesario.

Línea 910

Los valores de la coordenada Z para todos los puntos de la cara posterior están ajustados a ZZ/2.

Línea 930

La parte de la cara frontal del objeto que está conectada a los puntos análogos de la parte posterior está constituida por todos los puntos del primer segmento definido. Si sólo se define un segmento en total, la línea de comienzo del segundo segmento quedará indefinida, lo cual producirá un error en el programa en la línea 1020. Para evitarlo, S(1,2) se ajusta al valor de LB (número de líneas del primer segmento) si sólo hay un segmento.

Línea 940

El número de líneas es 2*LB + S(1,2)-1. En otras palabras, el doble de las líneas de la cara frontal, más el número de líneas del primer segmento definido (líneas que conectan las caras anterior y posterior).

Líneas 950-970

El primer conjunto de líneas a escribir en el fichero de datos son las líneas de la "superficie frontal". Estas son las líneas presentes en la versión SKETCH.

Líneas 980-1010

El segundo conjunto de líneas son las correspondientes de la cara posterior. Son las mismas que las de la cara frontal, además de otras **NA** líneas, siendo **NA** el número de puntos de la cara frontal.

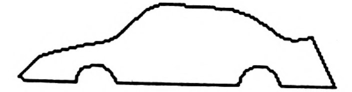
Líneas 1020-1050

Las líneas restantes son las líneas de conexión entre el primer segmento de la cara frontal y los puntos correspondientes de la cara posterior.

Líneas 1060-1100

La información sobre los segmentos se escribe en el fichero de datos.
La Figura 7.8 muestra la superficie frontal de una figura creada utilizando SKETCH3D para definir los datos. La figura tridimensional reconstruida se muestra en la Figura 7.12.

PROGRAMA SKETCH3D
EL VALOR DE X ES 340
EL VALOR DE Y ES 84
CUAL ES EL VALOR DE Z?



DISPARO=PRINC/FIN DE LA LINEA B=CORTE DE LINEA
F=FIN S=SIG SEGMENTO E=FIN DE SEGMENTO

Figura 7.8 Creación con SKETCH3D de la cara frontal de una imagen 3D.

7.4 Proyecciones paralelas

Las proyecciones paralelas más sencillas se llaman proyecciones ortográficas. Todos estamos familiarizados con las vistas de "planta", "alzado" y "perfil" de los planos arquitectónicos. Son proyecciones *ortográficas* perpendiculares a uno de los tres ejes, X,Y o Z. Estas vistas son muy aburridas en el caso de un objeto rectangular que descansa sobre los ejes principales, ya que no proporcionan ninguna ilusión de profundidad.

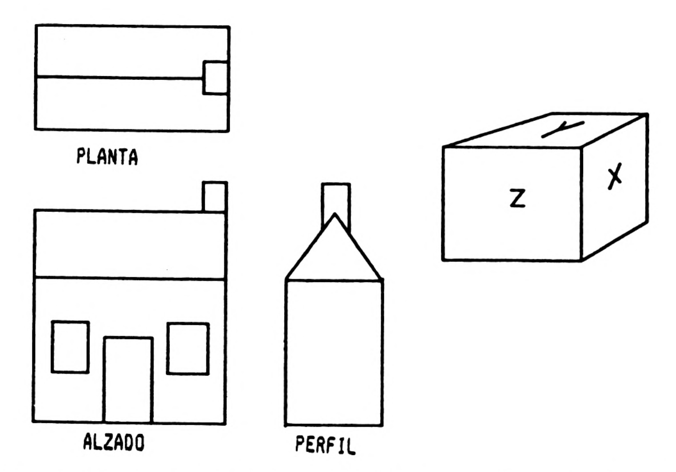


Figura 7.9 El tipo más sencillo de proyección: un diseño arquitectónico. La planta, el alzado y el perfil se corresponden con los planos perpendiculares a los ejes Z, Y y X.

Para programar proyecciones ortográficas de este tipo, lo único que hay que hacer es "olvidarse" del eje Z. Todo conjunto de datos tridimensionales representado como una figura bidimensional X,Y es una proyección ortogonal sobre el plano Z. Este tipo de programa nos presenta el formato que debe tener el código para manejar datos tridimensionales definiendo las diversas estructuras de datos (matrices X, Y y Z) necesarias para guardar los datos. El siguiente programa PROY3D apunta las directrices generales de las rutinas que desarrollaremos en este capítulo, que pueden ser aplicadas a cualquier conjunto de datos tridimensionales para dibujar una proyección paralela ortogonal.

Este programa emplea la misma sección de entrada de ficheros utilizada en otros programas, de modo que podemos emplear nuestros propios datos creados mediante FICHERO3D ó SKETCH3D, si es necesario. Si desea dejar la creatividad para más adelante, después del listado del programa se ofrecen algunas líneas alternativas, 60-130. Estas líneas incluyen sentencias DATA que representan un dibujo en pantalla de la casa de Caperucita.

Programa PROY3D

```
10 REM ****PROGAMA PROY3D****
20 REM Demostracion de un sencillo metodo de proyeccion en 2 dimensiones
30 DIM x(50),y(50),z(50),ln(2,50)
40 DIM xp(50),yp(50)
45 CLS
50 REM Toma los datos para dibujar la casa
60 OPENIN "casa.dat"
70 INPUT #9,npts
80   FOR i=1 TO npts
90     INPUT #9,x(i),y(i),z(i)
100    xp(i)=x(i)
110    yp(i)=y(i)
120  NEXT i
122  INPUT #9,li
124  FOR i=1 TO li
126    INPUT #9,ln(1,i),ln(2,i)
128  NEXT i
130  CLOSEIN
140  REM Dibujo de los ejes
150  MOVE 180,280
160  DRAW 180,180
170  DRAW 280,180
180  REM Ahora dibuja la casa
185    REM Observe las coordenadas son respecto al origen con una
traslacion de +200
190    FOR i=1 TO li
200      MOVE xp(ln(1,i))+200,yp(ln(1,i))+200
210      DRAW xp(ln(2,i))+200,yp(ln(2,i))+200
220    NEXT i
230  END
```

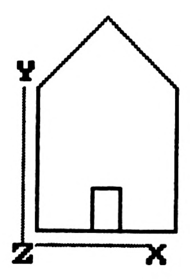


Figura 7.10 Representación con PROY3D de la fachada de la casa de Caperucita.

PROY3D es un programa muy simple, que consta de las siguientes secciones:

```
LINEAS 30- 40   DEFINICION DE MATRICES
       60-130   ENTRADA DE DATOS PARA LA CASA
      180-220   DIBUJO DE LA CASA
```

Líneas 30-40

Aunque no se efectuará ninguna transformación de este tipo en los datos de las matrices X e Y, se introducen las matrices XP e YP, ya que las utilizaremos extensivamente en posteriores capítulos para guardar los datos transformados.

Líneas 60-130

Los datos leídos son, como cabría esperar, idénticos a los datos bidimensionales, excepto por la adición de coordenadas Z. Los valores X e Y se almacenan también en las matrices XP e YP a medida que se van leyendo.

Líneas 140-170

Los ejes X e Y se dibujan para indicar la posición del origen desplazado con respecto al objeto. De nuevo, este código se incluye sólo por motivos de compatibilidad con otros programas posteriores.

Líneas 180-220

Aunque los datos son tridimensionales, esta sencilla proyección constituye una transformación de tres a dos dimensiones, omitiendo adecuadamente la coordenada Z asociada. Observe la forma de las sentencias MOVE y DRAW en las líneas 200-210. Los datos coordenados se dibujan alrededor del origen (es decir, el punto central "invisible" de la casa está en 0,0. Para trasladar la figura al centro de la pantalla, se suma a cada valor coordenado X e Y un valor de 200.

Estas son las sentencias necesarias para introducir directamente en el programa el trazado de la casa de Caperucita. Sustituya las líneas 50-130 por

```
50  rem  entrada de datos para la casa
60  npts=14
70  for i=1 to 14: read x(i),Y(i),z(i):xp=x(i):yp=y(i):next
80  lineas=18
90  for i=1 to lineas:read w(1,i),w(2,i):next i
100 data 0, 0, 0, 0, 45, 0, 22, 60, 0, 45, 45, 0, 45, 0, 0, 0, 0, 40, 0, 45,
    40, 22, 60, 40, 45, 45, 40, 45
110 data 0, 40, 15, 0, 0, 15, 21, 0, 30, 21, 0, 30, 0, 0, 1, 2, 2, 3, 3, 4,
    4, 5, 5, 1, 1, 6, 2, 7
115 data 3, 8, 4, 9, 5, 10, 6, 7, 7, 8, 8, 9, 9, 10, 10, 6, 11, 12, 12, 13,
    13, 14
```

Ahora, ya haya dibujado la casa de Caperucita o reconstruido su propio fichero mediante FICHERO3D, no tardará en darse cuenta de que el efecto de PROY3D es dibujar una vista sencilla y bastante aburrida del objeto. Del programa, tal y como está, no puede extraerse ninguna evidencia de tridimensionalidad. Para animar un poco el asunto es preciso transformar los datos.

7.5 Revisión de las rotaciones, traslaciones y cambios de escala

La verdadera utilidad de los gráficos en tres dimensiones se manifiesta únicamente cuando se va a trabajar de alguna forma sobre el dato tridimensional, ya sea haciéndolo girar, o empleando técnicas de reconstrucción con líneas ocultas. Por supuesto, la técnica concreta a utilizar dependerá de la aplicación. Los juegos de salón (tipo "*arcade*") o los simuladores de vuelo pueden necesitar bastantes rotaciones y traslaciones: el efecto de volar en una nave espacial alrededor de un obstáculo tridimensional, por ejemplo. Los programas de diseño asistido por ordenador pueden requerir que un objeto sea visualizado de la forma más realista posible, en cuyo caso será necesario que los objetos muestren superficies sólidas, eliminando las ocultas y creando efectos de sombreado.

Como hacíamos con el caso bidimensional en el Capítulo 4, comenzaremos por la rotación. Al igual que antes, el trabajo pesado lo realizaremos mediante álgebra de matrices, por lo que, una vez más, si desea conocer los tejemanejes de las técnicas que intervienen, consulte el Apéndice 2. En dos dimensiones nos bastaba especificar el origen y girar los puntos en torno a él. En tres dimensiones hemos de definir un eje de giro. Lo más sencillo es considerar la rotación en torno a uno de los ejes coordenados, X, Y o Z, por lo que comenzaremos considerando estas rotaciones "axiales". Construiremos un programa tridimensional equivalente a TRV. (descrito en el Capítulo 4), llamado TRASL3D. El código empleado en PROY3D es la base del de TRASL3D, y emplea una rutina de rotación para generar la matriz para las rotaciones en tres dimensiones.

Programa TRASL3D

```
10  REM **** PROGRAMA TRASL3D****
20  REM Demostracion de un sencillo metodo de proyeccion 3D
30  DIM x(30),y(30),z(30),ln(2,50),a(4,4)
40  DIM xp(30),yp(30)
50  CLS
55  MODE 1:INK 0,13:INK 1,1:INK 2,3
60  REM Toma los datos para dibujar la casa
70  OPENIN "casa.DAT"
80  INPUT #9,npts
90    FOR i=1 TO npts
100      INPUT #9,x(i),y(i),z(i)
105 IF z(i)>0 THEN z(i)=z(i)-60
110      xp(i)=x(i)
```

```
120       yp(i)=y(i)
130    NEXT i
140 INPUT #9,li
150    FOR i=1 TO li
160       INPUT #9,ln(1,i),ln(2,i)
170    NEXT i
180 CLOSEIN
182 REM toma el eje de giro
183    INPUT"Eje de giro? x=1, Y=2, Z=3";m
184    theta=0
222    GOSUB 400:REM Ajuste completo de la rotacion - solo la primera vez
225 REM Comienzo del bucle principal para la rotacion
227  k$=INKEY$:IF  k$="S"  OR  k$="s"  THEN  CLS:GOTO  182:REM  Abortar  la
rotacion
230 REM Ahora dibuja la casa
240 REM Las coordenadas son respecto al origen, con traslacion al centro
245 CLS
247 GOSUB 1000:REM Dibuja los ejes
250    FOR i=1 TO li
260       MOVE xp(ln(1,i))+320,yp(ln(1,i))+200
270       DRAW xp(ln(2,i))+320,yp(ln(2,i))+200,1,0
280    NEXT i
290 REM Giro de 10 grados
300    theta=theta+0.174533
310    a(m1,m2)=SIN(theta)
311    a(m1,m1)=COS(theta):a(m2,m2)=COS(theta):a(m2,m1)=-SIN(theta)
320 REM Calcula la proyeccion en el plano X,Y
330    FOR i=1 TO npts
340       xp(i)=a(1,1)*x(i)+a(1,2)*y(i)+a(1,3)*z(i)+a(1,4)
350       yp(i)=a(2,1)*x(i)+a(2,2)*y(i)+a(2,3)*z(i)+a(2,4)
360    NEXT i
370 REM Fin del bucle principal
380 GOTO 225
400 REM Subrutina de giro
410 c=COS(theta)
420 s=SIN(theta)
430    FOR k=1 TO 4
440    FOR l=1 TO 4
450      a(k,l)=0
460    NEXT l
470    NEXT k
475 REM Ahora calcula las inserciones adecuadas de la matriz
476 REM Vea la teoria en el Apendice!
480    a(4,4)=1
490    a(m,m)=1
500    m1=3-m:IF m1=0 THEN m1=1
510    m2=3:IF m=3 THEN m2=2
520    a(m1,m1)=c:a(m2,m2)=c:a(m2,m1)=-s:a(m1,m2)=s
530 RETURN
1000 REM Dibujo de los ejes
```

```
1010    MOVE 310,300
1020    DRAW 310,190,2,0
1030    DRAW 400,190,2,0
1032    LOCATE 20,6:PRINT"Y"
1034    LOCATE 26,14:PRINT"X"
1036    LOCATE 20,14:PRINT"Z"
1040 RETURN
```

```
LINEAS
  30-  40 DEFINICION DE MATRICES
  50-  55 DEFINICION DE LOS COLORES Y LA PANTALLA
  60- 180 ENTRADA DE DATOS PARA LA CASA
 182- 184 ENTRADA DEL EJE DE GIRO, INICIALIZACION DEL ANGULO DE GIRO
 222      GENERACION DE LA MATRIZ DE ROTACION
 225      COMIENZO DEL BUCLE PRINCIPAL
 250- 280 DIBUJO DE LA CASA
 290- 300 INCREMENTO DEL ANGULO, LLAMADA A LA SUBRUTINA DE ROTACION
 310- 311 AJUSTE DE LA MATRIZ DE ROTACION
 320- 360 CALCULO DE LA PROYECCION SOBRE EL PLANO X,Y
 370      FIN DEL BUCLE PRINCIPAL
 400- 530 SUBRUTINA DE GIRO
1000-1040 DIBUJO DE LOS EJES
```

Líneas 10-247

Esta sección del programa es idéntica, en gran medida, a la de PROY3D, salvo en que se elige un eje•de rotación M y se especifica el ángulo de rotación inicial.

Líneas 290-311

El ángulo de rotación se incrementa. La función trigonométrica de la subrutina de rotación trabaja en radianes, y el incremento es de 20 grados (.174533*2 radianes). Puesto que sólo unos pocos de los valores de la matriz se modifican después de cada incremento angular, se actualizan individualmente, en lugar de actualizar todos los de la matriz cada vez.

Líneas 320- 360

La proyección sobre el plano X,Y se lleva a cabo mediante álgebra de matrices. En este caso los multiplicandos son la matriz de rotación de 4 x 4 que se guarda en la tabla A y el vector de 4 x 1 que guarda las coordenadas X,Y y Z. La multiplicación de matrices trabaja con las matrices X,Y y Z, pero los resultados se ponen en las matrices XP e YP. Esto garantiza que cada multiplicación se lleva a cabo sobre los datos X,Y "vírgenes", evitando "rotaciones de las rotaciones".

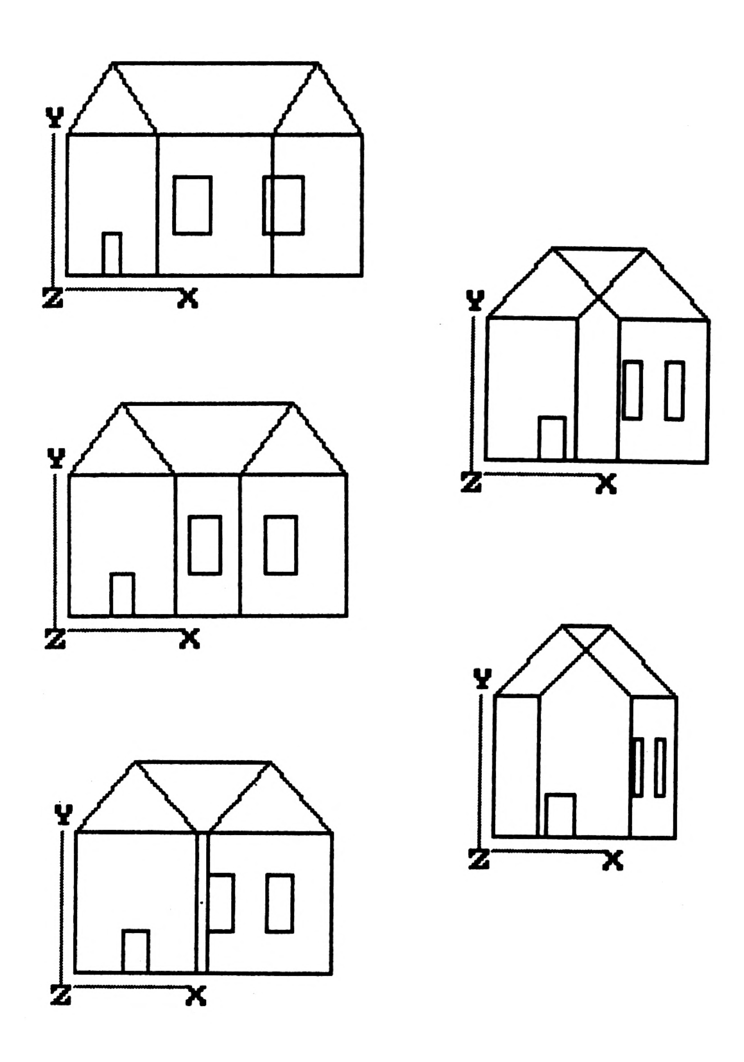


Figura 7.11 Resultado de TRAS3D que muestra la rotación de la casa de Caperucita en torno al eje Y.

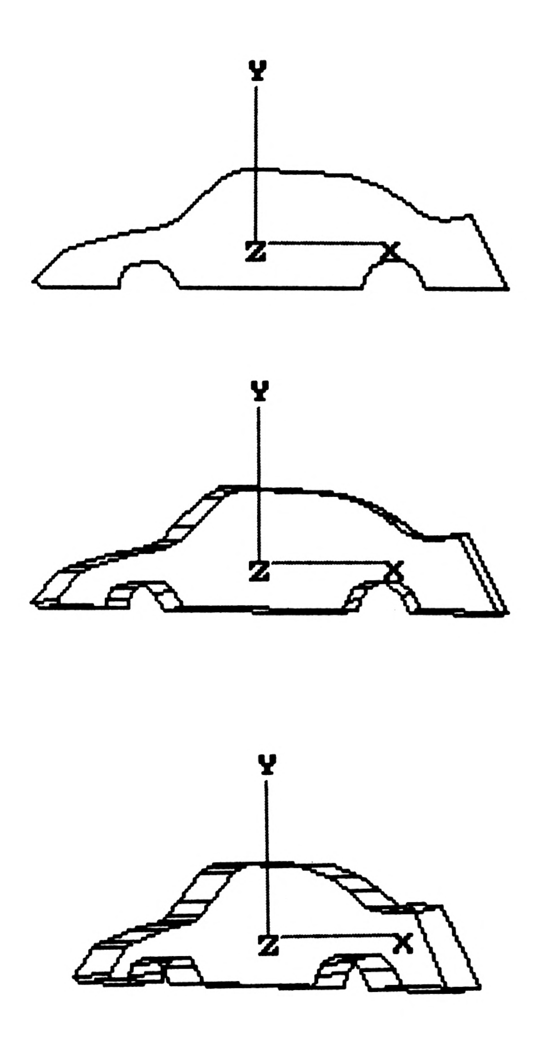


Figura 7.12 Resultado de TRASL3D que muestra la rotación del chasis de un coche. Esta figura se ha creado con SKETCH3D como se ve en la Figura 7.8.

Líneas 400-530

La subrutina de rotación no realiza ninguna multiplicación de matrices; simplemente coloca los valores correctos en la matriz de rotación A. La subrutina parece algo más complicada de lo que es en realidad, debido a que utiliza la variable **M** para seleccionar los elementos a sustituir por las diversas permutaciones de ángulos, ceros y unos. El Apéndice 2 nos muestra la forma de las tres matrices de rotación en torno a los tres ejes principales.
Si ejecutamos TRASL3D veremos girar el objeto en torno a uno de los ejes, X Y o Z. Hemos visto ya que las rotaciones son más sencillas si nos ceñimos a uno de los ejes principales. Los algoritmos para rotaciones más complejas pueden encontrarse en los libros sobre gráficos avanzados que se muestran en el Apéndice 3.

Tanto el cambio de escala como la traslación son eqivalentes a sus contrapartidas bidimensionales. La forma de definir y utilizar las matrices de las transformaciones es también similar en dos y en tres dimensiones. El fundamento matemático de las transformaciones tridimensionales aparece en el Apéndice 2, y las rutinas que deben añadirse a TRASL3D son las siguientes:

```
1200    REM SUBRUTINA DE CAMBIO DE ESCALA
1210    A(1,1)=SX:A(1,2)=0:A(1,3)=0:A(1,4)=0
1220    A(2,1)=0:A(2,2)=SY:A(2,3)=0:A(2,4)=0
1230    A(3,1)=0:A(3,2)=0:A(3,3)=SZ:A(3,4)=0
1240    A(4,1)=0:A(4,2)=0:A(4,3)=0:A(4,4)=1
1250    RETURN
1300    REM SUBRUTINA DE TRASLACIONES
1310    A(1,1)=1:A(1,2)=0:A(1,3)=0:A(1,4)=TX
1320    A(2,1)=0:A(2,2)=1:A(2,3)=0:A(2,4)=TY
1330    A(3,1)=0:A(3,2)=0:A(3,3)=1:A(3,4)=TZ
1340    A(4,1)=0:A(4,2)=0:A(4,3)=0:A(4,4)=1
1350    RETURN
```

Dejemos como ejercicio para el lector la modificación de los programas de este capítulo para utilizar estas rutinas.

7.6 Proyecciones en perspectiva

El algoritmo que utilizaremos para la proyección en perspectiva es muy simple, y se basa en el siguiente razonamiento: observe el diagrama que aparece a continuación. Puede verse que hay dos posibles tipos de planos de proyección: los que están frente al objeto y los que se encuentran detrás de él. Puede comprobarse también que el tamaño relativo de la proyección en comparación con el tamaño del objeto dependerá de la

posición del plano de perspectiva. Si se encuentra detrás del objeto, la proyección será más grande, pero si se encuentra delante del objeto será más pequeña. Observe también que cada punto del objeto se aplica en el punto correspondiente del plano de proyección. La imagen está formada, por supuesto, por líneas que unen los puntos proyectados, de la misma forma que si se tratase de un objeto puramente bidimensional (por eso nuestra matriz W se trata de la misma manera en los casos de dos y de tres dimensiones).

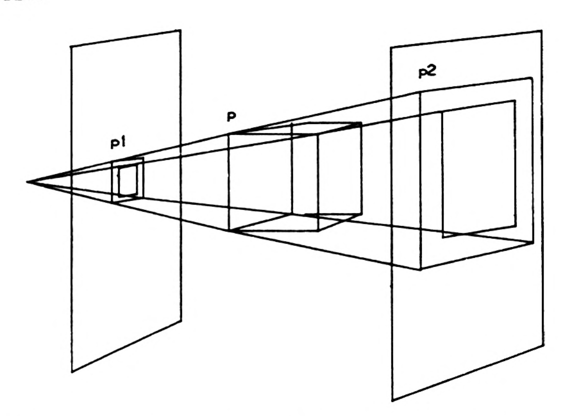


Figura 7,13 Proyección de un cubo en los planos anterior y posterior para mostrar la perspectiva. La cara p se proyecta en p1 (plano anterior), y en p2(plano posterior).

El siguiente diagrama nos muestra la relación geométrica que existe entre la posición del punto en el espacio de tres dimensiones y la posición de la proyección del mismo punto en la pantalla. Supongamos que el plano de proyección se encuentra a una distancia PP del ojo. Para todo punto X,Y,Z, debemos calcular los valores X,Y en el punto Z=PP. Para la Y, y/PP=Y/Z+PP, e y=Y*PP/Z+PP; del mismo modo, x=X*PP/Z+PP, con lo que los cálculos para la vista en perspectiva quedan concluidos.

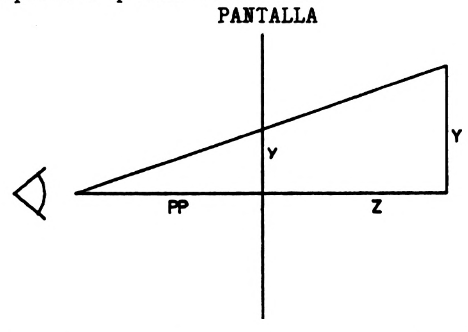


Figura 7,14 Relación entre la altura "real" de un objeto Y y su proyección en la pantalla. Conocida la distancia del observador a la pantalla (PP) y el valor Z del punto, puede calcularse *y* por semejanza de triángulos.

·Ahora que ya hemos realizado el trabajo duro, podemos sentarnos a contemplar las proyecciones en perspectiva en la pantalla. Como habrá deducido de la descripción anterior, no nos supondrá demasiado trabajo modificar TRASL3D para manejar proyecciones en perspectiva.

Programa PERS3D

```
10 REM**** PROGRAMA PERS3D ****
20 REM Demostracion de un metodo de proyeccion en perspectiva
30 DIM x(60),y(60),z(60),ln(2,80),a(4,4)
40 DIM xp(60),yp(60),zp(60)
50 CLS
55 MODE 1:INK 0,13:INK 1,1:INK 2,3
60 INPUT "NOMBRE DEL FICHERO";n$
70 OPENIN n$
80 INPUT #9,npts
90   FOR i=1 TO npts
100     INPUT #9,x(i),y(i),z(i)
105 IF z(i)>0 THEN z(i)=z(i)-60
110     xp(i)=x(i)
120     yp(i)=y(i)
125     zp(i)=z(i)
130   NEXT i
140. INPUT #9,li
150   FOR i=1 TO li
160     INPUT #9,ln(1,i),ln(2,i)
170   NEXT i
180 CLOSEIN
182 REM Toma el eje de giro
183   INPUT"Eje de giro? - x=1, y=2, z=3";m
184   theta=0
190 REM Toma la distancia del observador al origen
195     INPUT"Distancia al origen";pp
222     GOSUB 400
225 REM Comienzo del bucle principal de rotacion
227  k$=INKEY$:IF k$="S" OR k$="s" THEN CLS:GOTO 182:REM Abortar esta
rotacion
230 REM Ahora dibuja la casa
240  REM Las coordenadas son con respecto al origen, con traslacion al
centro
245 CLS
247 GOSUB 1000:REM Dibujo de los ejes
250   FOR i=1 TO li
260     MOVE xp(ln(1,i))+320,yp(ln(1,i))+200
270     DRAW xp(ln(2,i))+320,yp(ln(2,i))+200,1,0
280   NEXT i
290 REM Giro de 10 grados
300   theta=theta+0.174533
310   a(m1,m2)=SIN(theta)
311   a(m1,m1)=COS(theta):a(m2,m2)=COS(theta):a(m2,m1)=-SIN(theta)
```

```
312  REM Borrado de la ultima figura
313  GOTO 320
314    FOR i=1 TO li
316      MOVE xp(ln(1,i))+200,yp(ln(1,i))+200
317      DRAW xp(ln(2,i))+200,yp(ln(2,i))+200,1,1
318    NEXT i
320 REM Calculo de la proyeccion en el plano X,Y
330    FOR i=1 TO npts
340      xt=a(1,1)*x(i)+a(1,2)*y(i)+a(1,3)*z(i)+a(1,4)
350      yt=a(2,1)*x(i)+a(2,2)*y(i)+a(2,3)*z(i)+a(2,4)
352      zt=a(3,1)*x(i)+a(3,2)*y(i)+a(3,3)*z(i)+a(3,4)
354      dd=zt+pp:xp(i)=xt*pp/dd:yp(i)=yt*pp/dd:zp(i)=dd
360    NEXT i
370 REM Fin del bucle principal
380 GOTO 225
400 REM Subrutina de rotacion
410 c=COS(theta)
420 s=SIN(theta)
430   FOR k=1 TO 4
440   FOR l=1 TO 4
450     a(k,l)=0
460   NEXT l
470   NEXT k
475 REM Ahora calcula las inserciones correctas de la matriz
476 REM Vea la teoria en el Apendice!
480   a(4,4)=1
490   a(m,m)=1
500   m1=3-m:IF m1=0 THEN m1=1
510   m2=3:IF m=3 THEN m2=2
520   a(m1,m1)=c:a(m2,m2)=c:a(m2,m1)=-s:a(m1,m2)=s
530 RETURN
1000 REM Dibujo de los ejes
1010   MOVE 310,300
1020   DRAW 310,190,2,0
1030   DRAW 400,190,2,0
1032   LOCATE 20,6:PRINT"X"
1034   LOCATE 26,14:PRINT"Y"
1036   LOCATE 20,14:PRINT"Z"
1040 RETURN
```

```
LINEAS
  30-  40 DEFINICION DE MATRICES
  50-  55 DEFINICION DE LOS COLORES Y LA PANTALLA
  60- 180 ENTRADA DE DATOS PARA EL OBJETO
 182- 184 ENTRADA DEL EJE DE GIRO, INICIALIZACION DEL ANGULO DE GIRO
 190- 195 ENTRADA DE LA DISTANCIA DEL OBSERVADOR DESDE EL ORIGEN
 222      LLAMADA A LA SUBRUTINA DE ROTACION
 225      COMIENZO DEL BUCLE PRINCIPAL
 227      LA ENTRADA DE LA LETRA "S" ABORTA ESTA SERIE DE ROTACIONES
 247      LLAMADA A LA RUTINA DE DIBUJO DE LOS EJES
```

250- 280 DIBUJO DEL OBJETO
290- 300 **INCREMENTO DEL ANGULO, LLAMADA A LA SUBRUTINA DE ROTACION**
310- 311 **AJUSTE DE LA MATRIZ DE ROTACION**
320- 360 **CALCULO DE LA PROYECCION EN EL PLANO X,Y**
370 **FIN DEL BUCLE PRINCIPAL PARA ESTA ROTACION**
400- 520 SUBRUTINA DE ROTACION

Líneas 60-180

Los datos tridimensionales se leen igual que en el anterior programa de este capítulo.

Línea 195

La variable PP es la distancia del ojo al origen, como comentábamos anteriormente.

Líneas 320-360

La proyección sobre el plano X,Y arranca con el cálculo de los valores de las variables XT,YT y ZT por multiplicación de matrices, de forma similar al cálculo de los valores XP e YP en TRASL3D. En este punto, los datos proporcionan una proyección paralela, y en la línea 260 se lleva a cabo la transformación a perspectiva, calculando los valores XP e YP (perspectiva) multiplicando las proyecciones paralelas de XT e YT por el cociente entre la distancia del observador al origen (PP) y la distancia del observador a la coordenada Z del punto.

Puede verse el efecto que produce un cambio en la posición del plano de proyección, modificando el valor PP. Claramente, el tamaño de la Figura dependerá del valor de PP. Si consideramos que el área del plano de proyección es el área de la pantalla del monitor, entonces habrá valores de PP que reflejarán la distancia a la que nos encontremos de la pantalla. Las distancias más "realistas" estarán, probablemente, entre PP = 2*h y PP = 3*h, donde h es la altura de la pantalla en pixels. Así, PP valdrá normalmente entre 800 y 1200. Pero no tome como reglas estas palabras. Es mejor que pruebe Vd. mismo. Si se hace PP mucho mayor que 600, tendremos una proyección casi paralela. Si PP es menor que 400, veremos una imagen distorsionada.

Ahora que ya hemos observado todo el abanico de técnicas tridimensionales de que disponemos para dibujar figuras con estructura "de alambre", podemos experimentar con diferentes conjuntos de datos y proyecciones para familiarizarnos con los diversos métodos implicados. Es importante que comprenda estas técnicas básicas antes de proseguir con el capítulo siguiente, para poder concentrarse en los intrincados métodos de líneas ocultas.

Una forma especializada de "truco tridimensional" que puede realizar ya es la representación de un "tablón" *(billboard)*. En la jerga de los gráficos

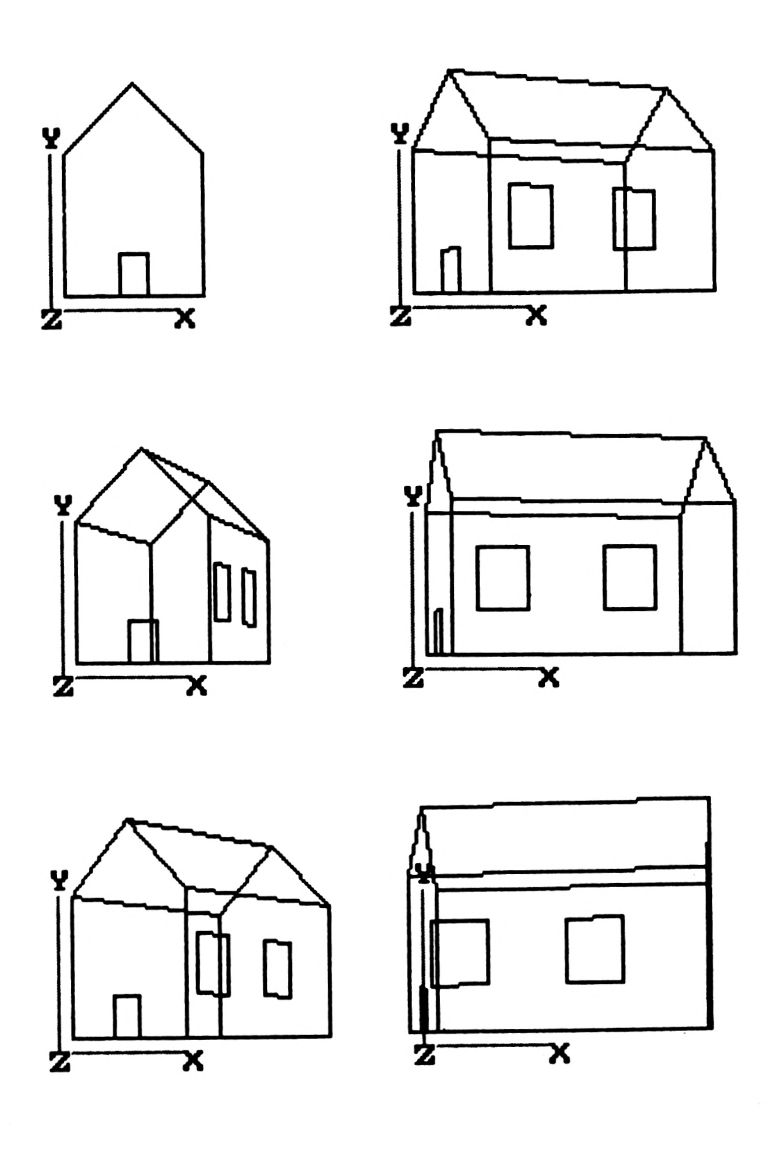


Figura 7,15 Resultado de PERS3D que muestra la rotación en perspectiva de la casa de Caperucita en torno al eje Y.

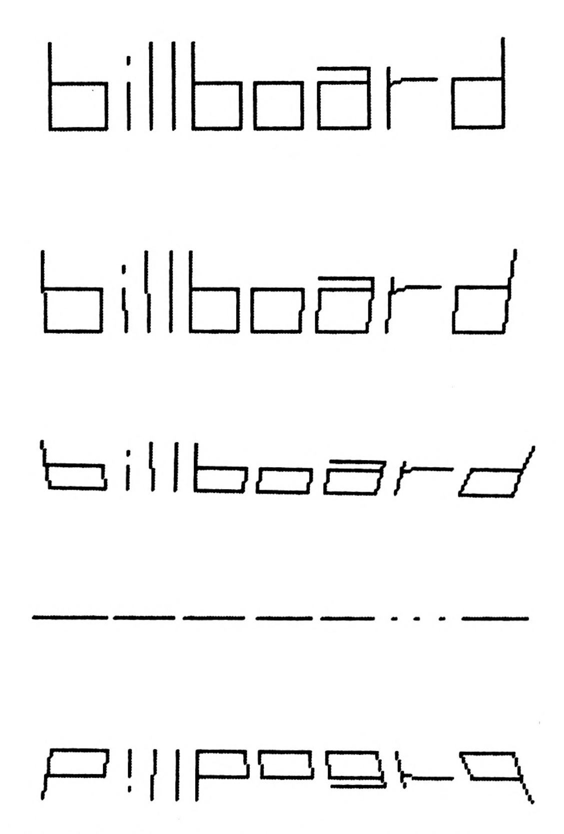


Figura 7.16 Utilización de la técnica del "tablón" para visualizar en tres dimensiones un plano bidimensional.

por ordenador, un tablón es un área de dibujo de dos dimensiones que se visualiza en tres dimensiones. Esta página del libro, por ejemplo, podría visualizarse en el monitor o en la TV como un plano de dos dimensiones, con la coordenada Z constante e igual a 0 para todo punto de la página. Pero ¿qué sucedería si girásemos la página, de forma que distintas partes de la misma tuvieran diferentes valores de Z, es decir, que unos puntos quedasen más cerca del observador que otros? Aquí empieza la diversión.

Con los listados de esta página, la creación de un tablón es muy simple. Basta modificar un programa como PER3D, por ejemplo, para que lea un conjunto de datos de dos dimensiones en lugar de datos tridimensionales. (Todos los datos creados por SKETCH podrían usarse, por tanto). A continuación, se crea un conjunto "artificial" de datos Z para compensar los datos que faltan en la serie bidimensional. Intente introducir estas alteraciones en PER3D.

```
10  REM *** TABLON ***
12  REM AMPLIACIONES DE PER3D PARA INTRODUCIR DATOS DE DOS DIMENSIONES
30  DIM X(500),Y(500),Z(500),LN(2,750),A(4,4)
40  DIM XP(500),YP(500),ZP(500)
100     INPUT #9,X(I),Y(I):Z(I)=0
```

¡Es así de simple!

La rotación de la figura hará rotar "automáticamente" el tablón en torno a los ejes X,Y o Z, cuando sea necesario. Las Figuras 7.16 y 7.17 nos muestran esta técnica en acción. Podrá reconocer en el conjunto de datos de la Figura 7.17 el mapa de los Estados Unidos del Capítulo 4. ¡Hemos creado una imagen "a vista de satélite"!

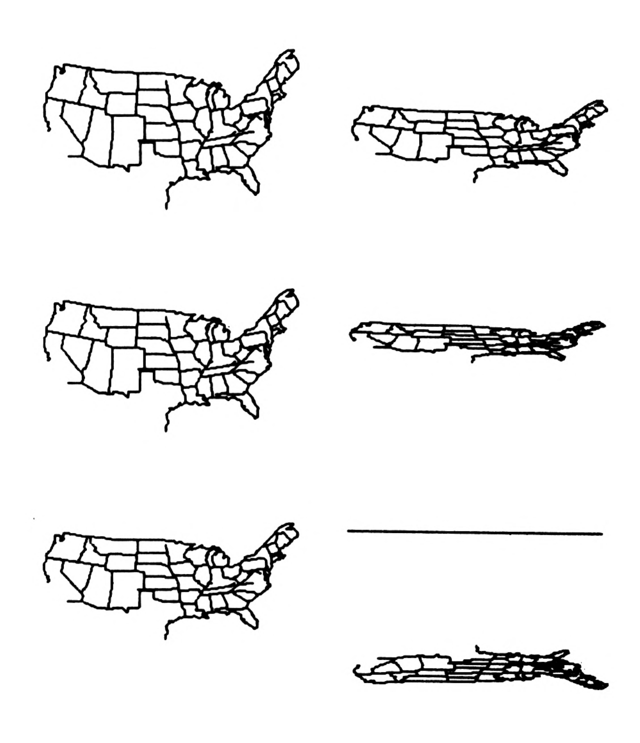


Figura 7.17 Los datos de la Figura 4.9 han servido para crear esta perspectiva de los Estados Unidos "a vista de satélite".

Capítulo 8

Líneas y superficies ocultas

8.1 ¿Qué es una línea oculta?

Todos los objetos tridimensionales que hemos considerado hasta ahora eran del tipo de "estructura de alambre". La casa de Caperucita que hacíamos girar en TRASL3D y PER3D aparecía de forma algo confusa, ya que a veces resultaba difícil determinar cuál era el frontal y cuál la parte posterior. Es preciso, por tanto, un método que "corte" las líneas que no deban ser vistas por el observador.

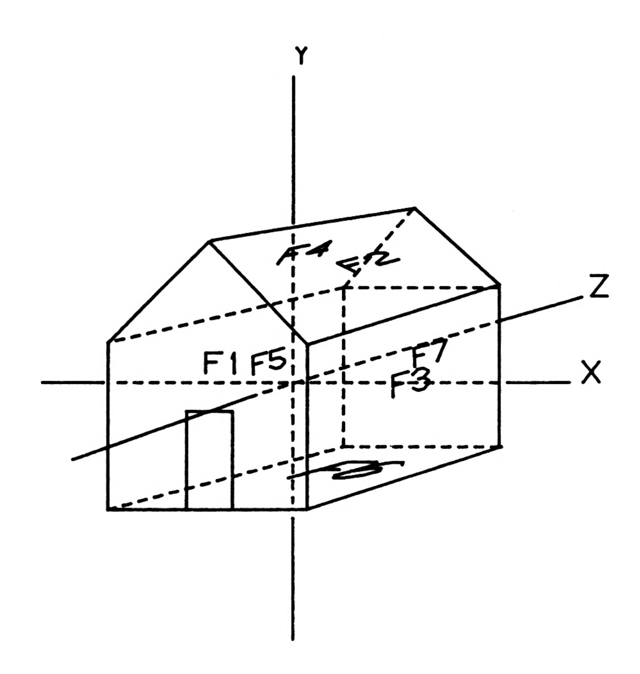


Figura 8,1 Trazado de la casa de Caperucita colocada alrededor del origen, Las líneas "ocultas" son las discontinuas, Las caras están codificadas como F1 - F7,

Estas líneas cortadas se llaman *líneas ocultas*, y se han desarrollado numerosos algoritmos para manejarlas. Algunos de estos métodos son relativamente simples, mientras que otros son mucho más complejos. El problema de todos estos algoritmos es la cantidad de tiempo de proceso que requieren, especialmente al programar en BASIC. En este capítulo examinaremos uno de los tipos más sencillos de algoritmos de líneas ocultas que podemos emplear en los microordenadores domésticos Amstrad.

Aunque es bastante habitual hablar de líneas ocultas, el término no es rigurosamente exacto. Lo que nos interesa en realidad no es saber qué *línea* pasa por delante de otra, sino qué *superficie*. Hasta ahora no hemos considerado superficies en absoluto, por lo que nuestra primera tarea en este capítulo será decidir qué entendemos por superficie, y cómo podemos describirla.

La forma más simple de ocultar líneas y superficies consiste en dibujar las superficies comenzando por la más alejada del observador, prosiguiendo hacia el "frente" de la pantalla. Siguiendo algunos criterios (el principal de los cuales ha de ser que todas las superficies deben dibujarse como sólidos, mediante algún método de coloreo o rellenado de superficies), puede producirse una imagen aceptable. Este método se llama "algoritmo del pintor", ya que cada superficie sucesiva "se pinta" sobre las subyacentes, de modo que la nueva superficie oscurece por completo o parcialmente a las que están por debajo de ella. He aquí un sencillo programa que dibuja cuatro rectángulos. El orden de los rectángulos, de abajo arriba, es: negro, azul, amarillo y verde, y la representación monocromática de la imagen producida se muestra en la Figura 8.2

Programa PINTOR

```
10 REM****PROGRAMA PINTOR****
20 REM Dibuja cuatro rectangulos para ilustrar el algoritmo del pintor
30 MODE 0
40 INK 0,13:INK 1,0:INK 2,2:INK 3,12:INK 4,9
50 REM DIBUJO DE LOS RECTANGULOS
60 REM Rectangulo negro
70 GRAPHICS PEN 1
80   FOR y=300 TO 200 STEP -1
90     MOVE 100,y
100    DRAW 400,y
110  NEXT y
120 REM Rectangulo azul
130 GRAPHICS PEN 2
140   FOR y=350 TO 250 STEP -1
150     MOVE 150,y
160     DRAW 350,y
170   NEXT y
180 REM Rectangulo amarillo
190 GRAPHICS PEN 3
200   FOR y=250 TO 150 STEP -1
```

```
210       MOVE 200,y
220       DRAW 400,y
230    NEXT y
240 REM Rectangulo verde
250 GRAPHICS PEN 4
260    FOR y=380 TO 200 STEP -1
270       MOVE 230,y
280       DRAW 500,y
290    NEXT y
299 |copy
300 END
```

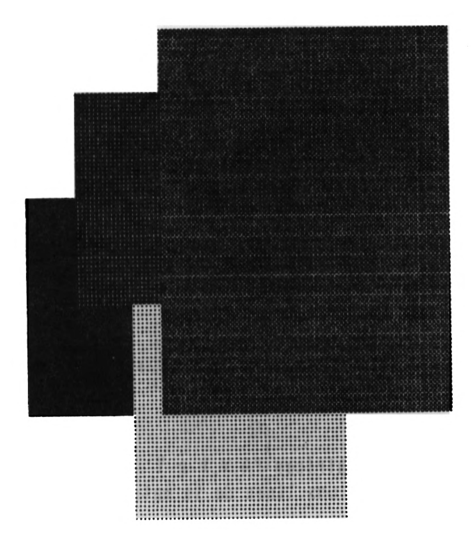


Figura 8.2 Utilización del "algoritmo del pintor" para dibujar superficies.

Este método para sobreescribir colores es la misma técnica empleada en el programa HISTO3D del Capítulo 5.

8.2 Definición de superficies

Repasemos nuestras estructuras para representar datos en tres dimensiones. Tenemos matrices X,Y y Z, que almacenan listas secuenciales de las coordenadas X,Y y Z de los puntos de la Figura. Tenemos también

una matriz W que contiene los datos necesarios para dibujar las líneas que unen los conjuntos de coordenadas. Necesitamos introducir ahora dos nuevas matrices que almacenarán la información de las superficies. Estas matrices se llaman FA y NL. FA se dimensiona como FA(i,j), donde i es el número de líneas que definen la superficie j. Los valores contenidos en cada elemento de la matriz serán los índices que guardamos en nuestra natriz W. Conociendo el valor de un determinado elemento de FA podemos, entonces, acceder a las coordenadas de los puntos de cualquiera de los extremos de la línea a la que se refiere. La segunda matriz, que se dimensiona como NL(k), guarda el número de líneas que definen cada superficie, donde k representa el número total de superficies.

Si esto no ha quedado claro, veamos el siguiente diagrama, que muestra la relación entre todas las matrices introducidas hasta ahora: los datos que se utilizan definen un cubo. Es importante que el lector se moleste en deducir esta relación, pues de lo contrario le será muy difícil crear sus propios conjuntos de datos para el tratamiento de líneas ocultas.

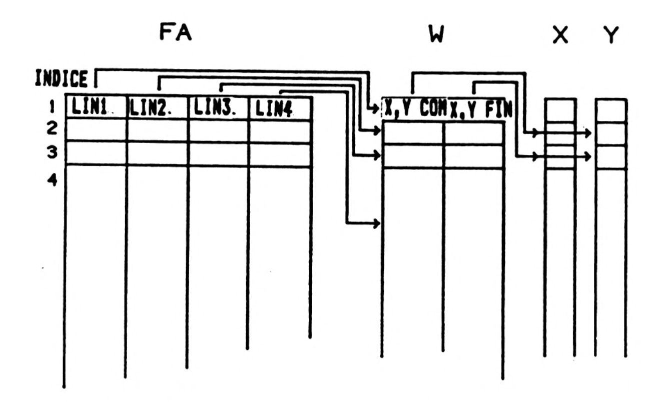


Figura 8,3 Relación entre las matrices FA,W,X e Y (comparar con la Figura 3,4). Cada elemento de la matriz FA apunta a una línea de la matriz W.

no.	*X*	*Y*	*Z*
1	−50	−50	−50
2	−50	50	−50
3	50	50	−50
4	50	−50	−50
5	−50	−50	50
6	−50	50	50
7	50	50	50
8	50	−50	50

líneas=12

no.	W 1	2
1	1	2
2	2	3
3	3	4
4	4	1
5	5	6
6	6	7
7	7	8
8	8	5
9	2	6
10	1	5
11	3	7
12	4	8

no.	NL
1	4
2	4
3	4
4	4
5	4
6	4

no.	FA 1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	3	11	7	12
4	1	9	5	10
5	4	10	8	12
6	2	9	6	11

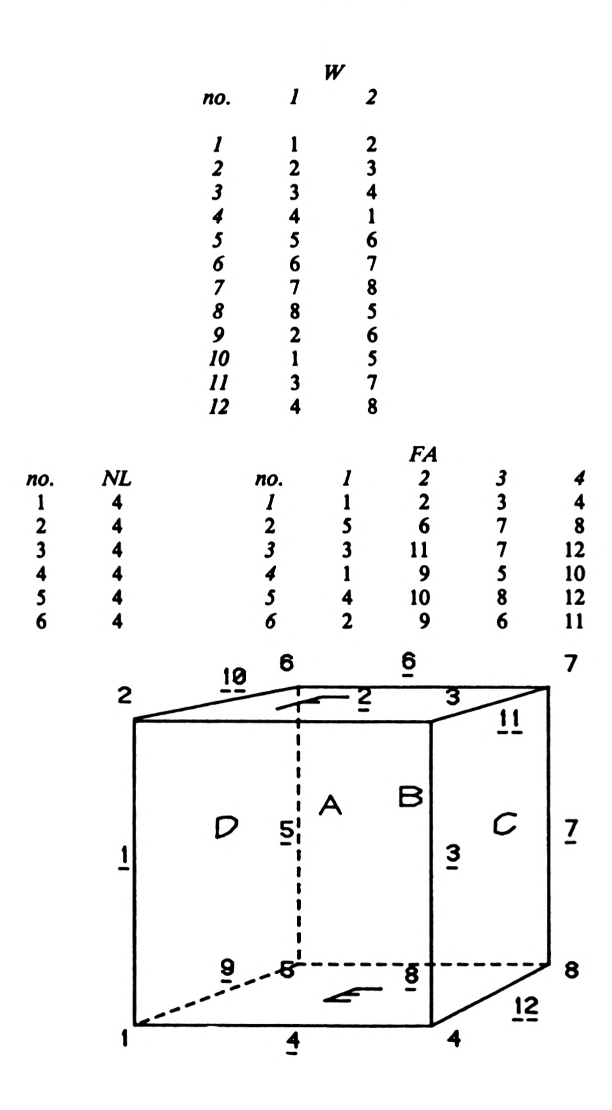


Figura 8.4 Puntos, líneas y caras de un cubo. Las líneas son las que están subrayadas. Las caras están etiquetadas con letras de la A a la F.

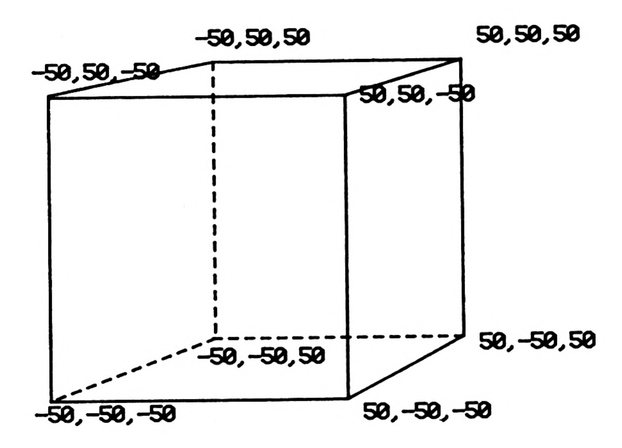


Figura 8.5 Datos coordenados del cubo utilizado en la demostración de líneas ocultas. Observe que el origen está dentro del cubo (a diferencia de la Figura 7.4).

Ya tenemos toda la información necesaria para definir cada superficie en términos de una malla poligonal, es decir, una serie cerrada de líneas que unen distintos puntos. Pero para poder efectuar el análisis de las superficies ocultas necesitamos también poder determinar qué superficies quedan delante de otras. En otras palabras, necesitamos conocer en qué plano del espacio de 3 dimensiones descansa cada superficie. Este procedimiento puede llevarse a cabo fácilmente mediante geometría del punto, y los métodos para ello pueden encontrarse en muchos libros de texto, algunos de los cuales se indican al final del libro como lecturas recomendadas para profundizar más en el tema.

8.3 Un programa completo de líneas ocultas

Armados con nuestras estructuras de datos ampliadas y con el algoritmo para calcular los coeficientes de los planos en el espacio de 3 dimensiones, podemos intentar desarrollar un algoritmo de líneas ocultas. Nuestro algoritmo sólo funcionará en un limitado número de casos. A pesar de todo, teniendo en cuenta las limitaciones de velocidad que nos impone el BASIC, se trata de una útil página de código. Comencemos exponiendo sus limitaciones. El algoritmo podrá emplearse si se cumplen las siguientes condiciones:

(1) Todas las superficies han de ser convexas (es decir, toda línea que se dibuje entre dos puntos cualesquiera de una superficie no adyacente ha de pertenecer a esa superficie).
(2) El origen (el punto 0,0,0) ha de estar situado en el interior del objeto. La razón de esto quedará aclarada más adelante.
(3) Sólo puede dibujarse un objeto.

Dentro de estas limitaciones, el programa es comparativamente rápido. Puede descomponerse en los siguientes pasos:

(1) Elegir tres puntos de cada superficie del objeto.
(2) Calcular el plano que contiene a esa superficie a partir de las coordenadas de esos tres puntos.
(3) Comprobar si el plano está entre el origen y el observador.
(4) Hacer una lista de todas las líneas de todos los planos que se encuentren entre el origen y el observador.
(5) Eliminar de la lista las duplicaciones.
(6) Dibujar las líneas para una figura con líneas ocultas.

Todos estos apartados parecen bastante inmediatos si se examinan por separado. Los cálculos más pesados tienen lugar en los pasos (4) y (5), ya que al procesar cada superficie y al añadir una nueva línea a la lista se produce redundancia, debido a que las líneas que están repetidas en varias superficies se repiten en la lista. Las entradas redundantes de la lista se eliminan (1) poniendo la lista en orden numérico y (2) suprimiendo las duplicaciones.

Podemos estar seguros, viendo la Figura 8.1, de que todas las superficies situadas entre Vd. (el observador) y el origen serán visibles, mientras que todas las demás no lo serán.

Los datos empleados en OCULTAS pueden crearse mediante FICHERO3D, versión de FICHERO2D que permite la incorporación de la información adicional necesaria para dibujar una figura con líneas ocultas, en concreto: el número de superficies, el número de líneas que limitan cada superficie y los códigos de esas líneas que permitirán "extraerlas" de las matrices existentes X,Y,Z y W. Este es el programa FICHERO3DO completo:

Programa FICHERO3DO

```
10 REM****PROGRAMA FICHERO3DO****
20 REM Programa para almacenar datos coordenados
25 CLS
30   INPUT"NOMBRE DEL FICHERO";h$
40   OPENOUT h$
50   INPUT"NUMERO DE PUNTOS";npts
55   WRITE #9,npts
60   PRINT "INTRODUZCA LAS TERNAS X,Y,Z"
70  FOR i=1 TO npts
80    INPUT"X,Y,Z=";x,y,z
90    WRITE #9,x
100   WRITE #9,y
105   WRITE #9,z
110  NEXT i
120 INPUT"NUMERO DE LINEAS";li
130  WRITE #9,li
140  PRINT"Numero de puntos de union"
150 FOR i=1 TO li
```

```
160    INPUT "numeros de los puntos de comienzo,final";sn,fi
170    WRITE #9,sn
180    WRITE #9,fi
190 NEXT i
200 REM Ahora toma los datos de las superficies.
210    INPUT"Numero de superficies";nf
215    PRINT #9,nf
220 REM  Dimensionamiento de las matrices de superficie
230    DIM fa(12,nf),nl(nf)
240     PRINT"Escriba las lineas que limitan cada superficie, en sentido
horario"
260    FOR i=1 TO nf
270      INPUT "Numero de lineas de esta superficie";nlf
275        nl(i)=nlf
280        FOR j=1 TO nlf
290          INPUT"Linea=";lnum
300          fa(j,i)=lnum
310          PRINT #9,lnum
320        NEXT j
330 REM Ahora llena la matriz con ceros hasta el numero maximo de linea
340        IF nlf=12 THEN 390
350        FOR k=j TO 12
360          fa(k,i)=0
370          PRINT #9,0
380        NEXT k
390   NEXT i
400 REM Ahora almacena la matriz de "lineas por superficie"
410  FOR i=1 TO nf
420    PRINT #9,nl(i)
430  NEXT i
440 CLOSEOUT
450 END
```

También se reproduce a continuación el programa OCULTAS completo, para evitar confusiones al adaptar TRASL3D, en cuya estructura está basado. Por su propia conveniencia, seguramente le será más fácil cargar TRASL3D y editar de forma extensiva que teclear todo el programa desde el principio.

Programa OCULTAS

```
5 REM****PROGRAMA OCULTAS****
10 REM Utilice PP=1000 para probar
20 MODE 1:INK 0,13:INK 1,1:INK 2,3:CLS
30 pp=1000
40 DIM x(40),y(40),z(40),ln(2,60),a(4,4),fa(12,40),nl(40),ls(40)
50 DIM xp(40),yp(40),zp(40)
60 REM Toma los datos para dibujar la superficie
70 INPUT"NOMBRE DEL FICHERO";h$:OPENIN h$
80      INPUT #9,npts
```

```
90      FOR i=1 TO npts
100       INPUT #9,x(i)
110       INPUT #9,y(i)
120       INPUT #9,z(i)
130       xp(i)=x(i):yp=y(i)
140     NEXT i
150   INPUT #9,li
160   FOR i=1 TO li
170     INPUT #9,ln(1,i),ln(2,i)
180   NEXT i
190   INPUT #9,nf
200   FOR i=1 TO nf
210     FOR j=1 TO 12
220       INPUT #9,fa(j,i)
230     NEXT j
240   NEXT i
250   FOR i=1 TO nf
260     INPUT #9,nl(i)
270   NEXT i
280   CLOSEIN
300 REM Toma el eje de giro
310   INPUT"Eje de giro? X=1, Y=2, Z=3";m
320   theta=0
330   GOSUB 540:REM Ajuste completo de la rotacion - solo la primera vez
340 REM Comienzo del bucle de giro principal
350    k$=INKEY$:IF k$="S" OR k$="s" THEN CLS:GOTO 300:REM Aborta esta
rotacion
360 REM Ahora dibuja el objeto
370 REM Las coordenadas son respecto al origen, con traslacion al centro
380   CLS
390   GOSUB 700
400   REM Giro de 10 grados
410     theta=theta+0.174533
420     a(m1,m2)=SIN(theta)
430     a(m1,m1)=COS(theta):a(m2,m2)=COS(theta):a(m2,m1)=-SIN(theta)
440 REM Calculo de la proyeccion en el plano X,Y
450     FOR i=1 TO npts
460       xt=a(1,1)*x(i)+a(1,2)*y(i)+a(1,3)*z(i)+a(1,4)
470       yt=a(2,1)*x(i)+a(2,2)*y(i)+a(2,3)*z(i)+a(2,4)
480       zt=a(3,1)*x(i)+a(3,2)*y(i)+a(3,3)*z(i)+a(3,4)
490       dd=zt+pp:xp(i)=xt*pp/dd:yp(i)=yt*pp/dd:zp(i)=zt
500     NEXT i
510 GOSUB 780:REM Rutina de lineas ocultas
520 REM Fin del bucle principal
530 GOTO 340
540 REM Subrutina de giro
550 c=COS(theta)
560 s=SIN(theta)
570    FOR k=1 TO 4
580    FOR l=1 TO 4
```

```
590      a(k,l)=0
600    NEXT l
610    NEXT k
620 REM Ahora calcula las inserciones adecuadas en la matriz
630 REM Vea la teoria en el Apendice!
640 a(4,4)=1
650 a(m,m)=1
660 m1=3-m:IF m1=0 THEN m1=1
670 m2=3:IF m=3 THEN m2=2
680   a(m1,m1)=c:a(m2,m2)=c:a(m2,m1)=-s:a(m1,m2)=s
690 RETURN
700 REM Dibujo de los ejes
710   MOVE 310,300
720   DRAW 310,190,2,0
730   DRAW 400,190,2,0
740   LOCATE 20,6:PRINT"Y"
750   LOCATE 26,14:PRINT"X"
760   LOCATE 20,14:PRINT"Z"
770 RETURN
780 REM Subrutina OCULTAS
790    ic=0:c=0:REM Inicializacion de los contadores
800 REM Toma de puntos del plano
810    FOR ih=1 TO nf
820      i1=fa(1,ih):i2=fa(2,ih)
830      i5=ln(1,i1):i6=ln(2,i1):i7=ln(1,i2)
840      IF (i5=i7) OR (i6=i7) THEN i7=ln(2,i2)
850 REM Calculo de la posicion del plano
860      x5=xp(i5)-xp(i6):y5=yp(i5)-yp(i6):z5=zp(i5)-zp(i6)
870      x6=xp(i7)-xp(i6):y6=yp(i7)-yp(i6):z6=zp(i7)-zp(i6)
880      a9=y5*z6-y6*z5
890      b9=z5*x6-z6*x5
900      c9=x5*y6-x6*y5
910      d9=a9*xp(i5)+b9*yp(i5)+c9*zp(i5)
920 REM Estan el observador y el origen en distintos lados del plano?
925      IF d9=0 THEN f9=0:GOTO 940:REM Previene la division por cero
930      f9=(1+c9*pp)/d9
940      IF f9>=0 THEN 1010
950 c=c+1
960 ix=nl(ih)
970      FOR jh=1 TO ix
980        ic=ic+1
990        ls(ic)=fa(jh,ih)
1000     NEXT jh
1010   NEXT ih
1020 REM Ordenamiento de la lista
1030   FOR ih=1 TO ic-1
1040     ii=ih+1
1050     ll=ls(ih)
1060     FOR jh=ii TO ic
1070     IF ll<=ls(jh) THEN 1110
```

```
1080      ll=ls(jh)
1090      ls(jh)=ls(ih)
1100      ls(ih)=ll
1110    NEXT jh
1120    NEXT ih
1130 REM Seguimiento de las duplicaciones de la lista
1140    jh=1
1150    FOR ih=2 TO ic
1160      IF ls(ih)=ls(jh) THEN 1190
1170      jh=jh+1
1180      ls(jh)=ls(ih)
1190    NEXT ih
1200    ic=jh
1210    in=1
1220    lq=ls(1)
1230 REM Ahora dibuja la figura utilizando solo las lineas de la lista
1240    FOR ih=1 TO li
1250      l2=ln(2,ih)
1260      l1=ln(1,ih)
1270    IF ih<>lq OR in>ic THEN 1320
1280      MOVE xp(l1)+320,yp(l1)+200
1290      DRAW xp(l2)+320,yp(l2)+200,1,0
1300      in=in+1
1310      lq=ls(in)
1320    NEXT ih
1330 FOR i=1 TO 1000:NEXT i
1340 RETURN
```

OCULTAS consta de las siguientes secciones:

LINEAS

20	SELECCIONA MODO Y COLORES
30	DEFINE LA DISTANCIA DEL OBSERVADOR AL ORIGEN
40- 50	DEFINE LAS MATRICES
70- 280	ENTRADA DE DATOS DEL OBJETO
300- 320	ENTRADA DE LOS EJES DE GIRO E INICIALIZACION DEL ANGULO DE GIRO
330	LLAMADA A LA SUBRUTINA QUE GENERA LA MATRIZ DE ROTACION
340	COMIENZO DEL BUCLE PRINCIPAL
350	SALIDA DE ESTA SECUENCIA DE ROTACION A TEXTO
390	LLAMADA A LA SUBRUTINA QUE DIBUJA LOS EJES
400- 430	INCREMENTO DEL ANGULO, ACTUALIZACION DE LA MATRIZ
440- 500	CALCULO DE LA PROYECCION SOBRE EL PLANO X,Y
510	LLAMADA A LA SUBRUTINA DE LINEAS OCULTAS
530	FIN DEL BUCLE PRINCIPAL PARA ESTA ROTACION
540- 690	SUBRUTINA DE GIRO (COMO EN EL CAPITULO 7)
700- 770	SUBRUTINA QUE DIBUJA LOS EJES
780	COMIENZO DE LA SUBRUTINA DE LINEAS OCULTAS
790	INICIALIZACION DE LOS CONTADORES

800- 840	LOS TRES PUNTOS NECESARIOS PARA CADA PLANO
850- 910	CALCULO DE LOS COEFICIENTES DE CADA PLANO EN EL ESPACIO DE 3 DIMENSIONES
920- 940	COMPROBACION DE SI EL OBSERVADOR Y EL ORIGEN SE ENCUENTRAN EN LADOS OPUESTOS DEL PLANO
950-1000	ALMACENAMIENTO DE LOS INDICES DE TODAS LAS LINEAS DE ESTE LADO EN LA MATRIZ LS
1010	PROCESAMIENTO DEL SIGUIENTE PLANO
1020-1120	ORDENACION DE LA LISTA LS
1130-1190	PREVENCION DE LAS DUPLICACIONES EN LA LISTA
1200-1220	INICIALIZACION DE VARIABLES PARA EL DIBUJO DE LA FIGURA
1230-1320	DIBUJO DE LA FIGURA MEDIANTE LAS LINEAS DE LA MATRIZ LS
1330	BUCLE DE RETARDO
1340	VUELTA AL PROGRAMA PRINCIPAL

Líneas 10-770

Todos estos pasos son parecidos a los que encontrábamos en el capítulo anterior para el programa PER3D. Esta parte del programa dibuja y gira un cubo en perspectiva en torno a uno de los ejes principales. La sección de entrada de los datos del objeto (líneas 70-280) se ha ampliado para leer los datos de las superficies, que se almacenarán en las matrices FA y NL. La sección de dibujo de los ejes (subrutina comprendida entre las líneas 700-770) sólo es ilustrativa, y puede omitirse si se desea.

Línea 790

Los dos contadores IC y C se inicializan. C es el número de superficies a visualizar después de la eliminación de las líneas ocultas. IC es el número de líneas de la lista LS.

Líneas 800-840

El método para determinar qué superficies son visibles se basa en el cálculo del "coeficiente del plano" para cada cara. Este coeficiente se determina a partir de las posiciones de tres puntos de cada cara. Las dos primeras líneas de cada cara se obtienen de la matriz FA (y se colocan en las variables I1 e I2).

Los puntos de comienzo de la primera línea y de la segunda se obtienen a continuación de la matriz V, y se colocan en las variables I5 e I6. El punto final de la primera línea se guarda en la variable I7. Si I5 ó I6 = I7, entonces el punto final de la segunda línea se coloca en I7.

Líneas 850-910

La posición del plano se calcula de la siguiente forma. Las diferencias entre las coordenadas X,Y y Z de los puntos I5 e I6 y entre las de los puntos I6 e I7 se colocan en las variables X5,Y5,Z5 y X6,Y6,Z6.

A continuación se calculan las variables A9, B9 y C9, y a partir de estos valores se halla el de la variable D9. Seguidamente, en la línea 640, se calcula el coeficiente del plano, F9. Si este coeficiente es menor o igual que 0.0, significa que la superficie no está entre el observador y el origen, con lo cual ésta se descarta.

Líneas 950-1000

Todas las líneas de la superficie (si es que no ha sido descartada) han de almacenarse en una matriz que se utilizará más adelante para dibujar la imagen final. Se incrementa el contador C (una superficie más), y el valor de IX se hace igual al número de líneas de la superficie (utilizando como referencia la matriz NL: observe que se asigna a la variable IH del bucle el valor del número de superficies). Para cada nueva línea se incrementa el valor del contador IC, y la matriz FA se emplea para obtener cada una de las sucesivas líneas de la superficie que está siendo procesada. Estas líneas se colocan a continuación en la matriz LS. Observe también que los índices de las líneas que se guardan en LS no están agrupados específicamente en superficies, sino que LS contiene únicamente una lista de líneas.

Líneas 1020-1190

La etapa siguiente, que se lleva a cabo una vez procesadas todas las superficies, supone, en primer lugar, ordenar la matriz LS, de modo que queden agrupadas todas las "repeticiones" (es decir, todas las apariciones de la misma línea en dos caras visibles). Una vez efectuada esta ordenación (líneas 1020-1120), es más fácil eliminar todas las repetiones de la misma recta (líneas 1130-1190). Dejo para el lector la tarea de averiguar cómo se realizan estos dos pasos.

Líneas 1200-1220

Debido a la eliminación de las duplicaciones, IC ya no es un verdadero indicador del número de líneas. En la línea 840 podemos ver que JH se utiliza para almacenar temporalmente el número "reducido" de líneas. Así pues, IC ha de tomar el valor de JH. La variable LQ sirve para almacenar el número de la línea actual (a partir de la lista LS), e IN es un contador para la lista LS.

Líneas 1230-1320

La figura se dibuja de forma esencialmente idéntica a la empleada para PER3D en el capítulo anterior. La principal diferencia es que LQ solamente cambia una vez alcanzada la línea "visible" actual en la lista principal de líneas. Si la línea que está siendo procesada actualmente es LQ, se dibujará esta línea, se incrementará IN y se asignará a LQ el valor del siguiente elemento de LS. En otras palabras, LQ se convierte en el siguiente número de línea a dibujar.

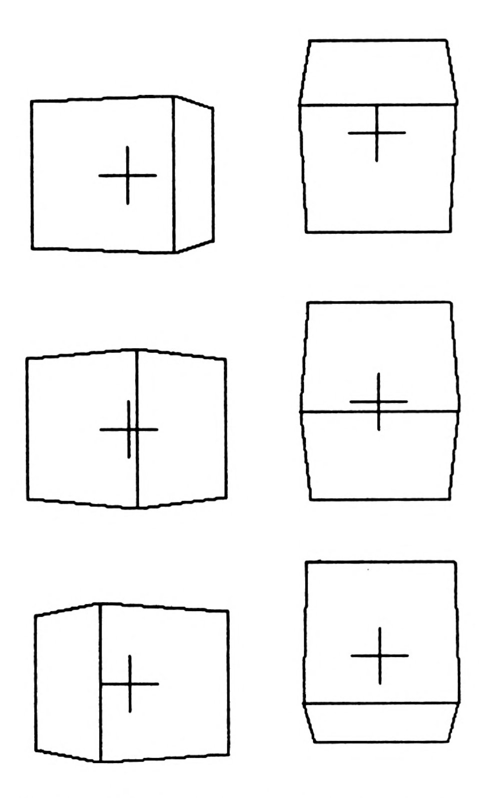


Figura 8.6 Resultado de OCULTAS que muestra la rotacion de un cubo alrededor de los ejes X e Y.

Puede utilizarse OCULTAS para cualquier conjunto de datos tridimensionales que no determinen un cuerpo cóncavo. Por supuesto, lo que esto implica es, simplemente, que la coordenada 0,0,0 ha de estar en el interior de la figura. Pero no hay que pensar que por ello estamos limitados a dibujar la figura en la esquina inferior izquierda de la pantalla. Lo que sucede es que deben emplearse las rutinas de traslación y cambio de escala, después del procesamiento de las líneas ocultas, para colocar el dibujo en el lugar deseado de la pantalla. Existe una forma de abreviar este proceso de desplazamiento de la figura, que ya empleábamos en OCULTAS. Todo lo que hacíamos era sumar 30 a los valores finales de las coordenadas X e Y a la hora de dibujar. Este incremento es completamente arbitrario, y puede tomar cualquier valor entre 0 y 639 para la X y entre 0 y 399 para la Y, dependiendo de la posición original de la figura a transformar.

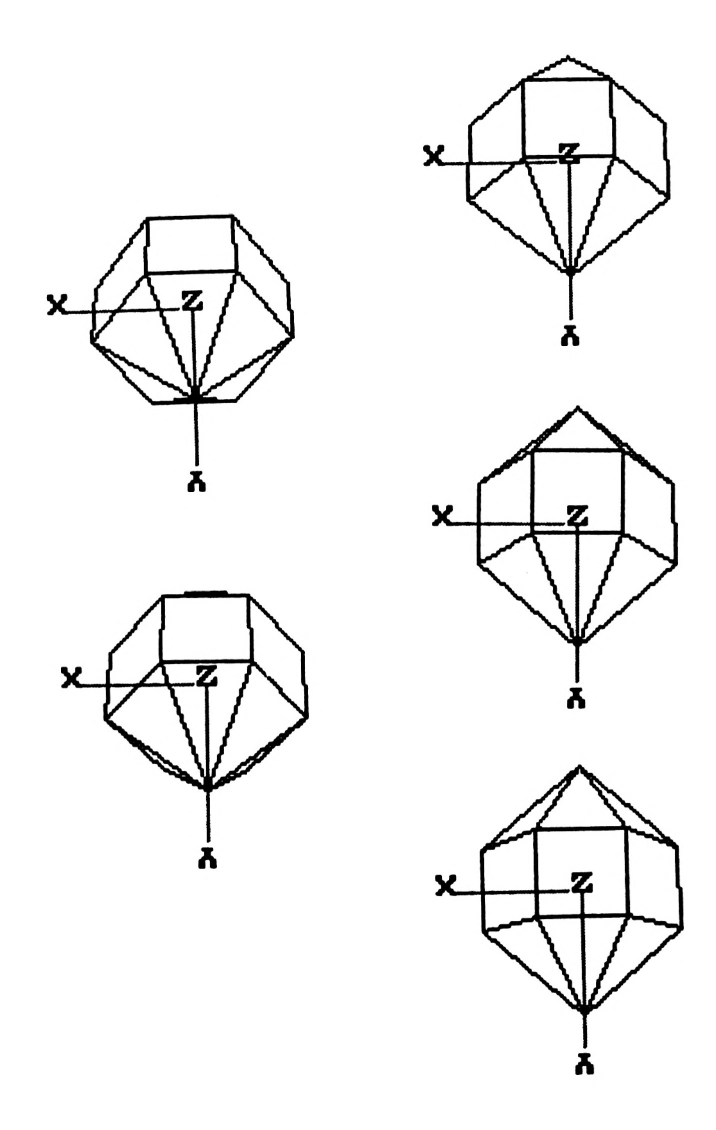


Figura 8.7 Resultado de OCULTAS que muestra la representación con líneas ocultas de un "cristal". Las rotaciones son con respecto a los ejes X,Y y Z (columnas izquierda, central y derecha, respectivamente).

La siguiente figura puede dibujarse mediante el conjunto de datos que se ofrece a continuación. Aunque su trazado es más complejo que el de un vulgar cubo, sigue verificando las condiciones que necesita para funcionar nuestro sencillo algoritmo de líneas ocultas.

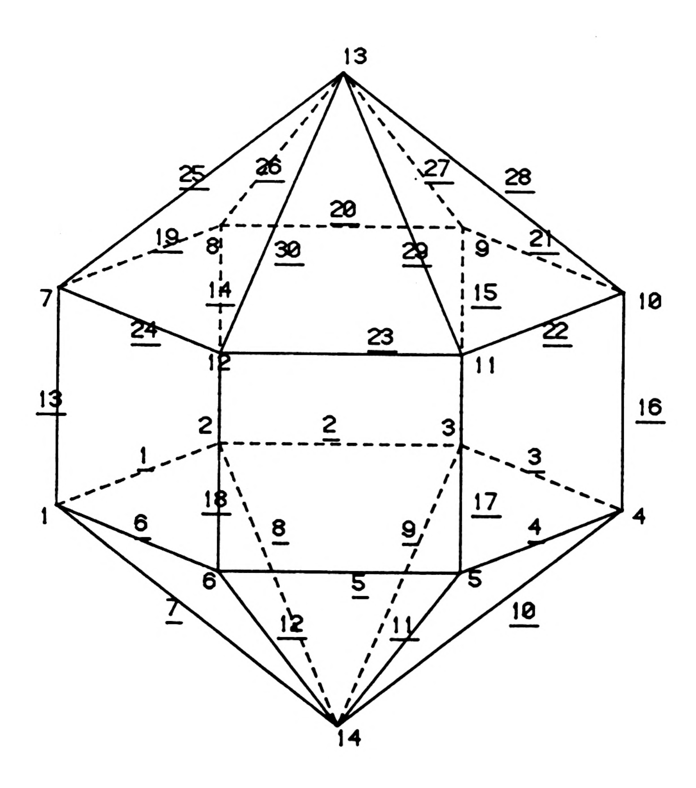


Figura 8,8 Datos de puntos y líneas para un cristal poliédrico de 14 caras, Los datos de superficies se han omitido de la figura para mayor claridad (ver los datos a continuación),

npts = 14

no.	*X*	*Y*	*Z*
1	100	70	0
2	140	70	−60
3	200	70	−60
4	240	70	0

5	200	70	60
6	140	70	60
7	100	130	0
8	140	130	−60
9	200	130	−60
10	240	130	0
11	200	130	60
12	140	130	60
13	170	190	0
14	170	10	0

lineas=30

	W	
no.	1	2
1	*1*	*2*
2	2	3
3	3	4
4	4	5
5	5	6
6	6	1
7	1	14
8	2	14
9	3	14
10	4	14
11	5	14
12	6	14
13	1	7
14	2	8

	W	
no.	1	2
15	3	9
16	4	10
17	5	11
18	6	12
19	7	8
20	8	9
21	9	10
22	10	11
23	11	12
24	12	7
25	7	13
26	8	13
27	9	13
28	10	13
29	11	13
30	12	13

			FA		
no.	*NL*	*1*	*2*	*3*	*4*
1	4	13	1	14	19
2	4	14	2	15	20
3	4	15	3	16	21
4	4	16	4	17	22
5	4	17	5	18	23
6	4	18	6	13	24
7	3	19	26	25	0
8	3	20	27	26	0
9	3	21	28	27	0
10	3	22	29	28	0

11	3	23	30	29	0
12	3	24	25	30	0
13	3	1	7	8	0
14	3	2	8	9	0
15	3	3	9	10	0
16	3	4	10	11	0
17	3	5	11	12	0
18	3	6	12	7	0

8.4 Extensión de SKETCH3D

La extensión de SKETCH que permite componer en pantalla conjuntos de datos de tres dimensiones puede ampliarse de nuevo para que incluya datos relativos a superficies. A continuación se ofrecen las líneas adicionales que deben añadirse a SKETCH3D para configurar esta nueva versión, a la que llamaremos SKETCH3DO. Los datos de las superficies están constituidos por el trazado inicial (superficie uno), el duplicado de este trazado (superficie dos) y las otras superficies, cada una de las cuales es un rectángulo formado por una línea de la superficie 1, su duplicado en la superficie 2 y las líneas que unen los puntos situados en ambos extremos de las dos líneas de ambas superficies. En la siguiente figura se indican las líneas de estas superficies.

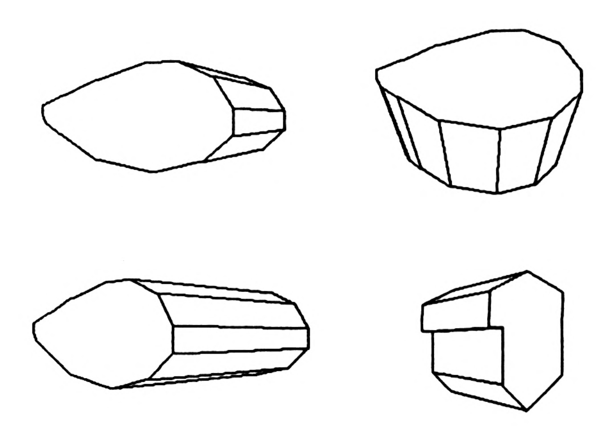


Figura 8,9 Resultado de SKETCH3DO, La figura de la esquina inferior derecha nos demuestra que, en contados casos, incluso una figura no convexa puede dibujarse con OCULTAS,

El programa principal OCULTAS necesita ciertas modificaciones para funcionar adecuadamente con SKETCH3D. Estas alteraciones consisten, fundamentalmente, en calcular el punto central de los datos, trasladando este punto al origen. Todos los demás puntos sufrirán ese mismo desplazamiento negativo. Recordemos que esta técnica se utilizó para transformar los datos de un segmento en el programa DISEÑO (Capítulo 6). Sólo es necesario transformar las coordenadas X e Y, ya que SKETCH3DH ajusta automáticamente los valores de Z de las superficies frontal y posterior de modo que sean mayor y menor que 0, respectivamente. Estos son los cambios que deben efectuarse en SKETCH3D.

Programa SKETCH3DO

```
10 REM****PROGRAMA SKETCH3D****
110 s(1,2)=0
800 REM Ahora crea un fichero que contiene los datos
810 OPENOUT n$
815 GOSUB 1200:REM Toma los valores de traslacion
817 PRINT"la cuantia del desplazamiento en X es",xh-xl
818 PRINT"la cuantia del desplazamiento en Y es",yh-yl
819 INPUT"introduzca la cuantia del desplazamiento en Z",zz
820   PRINT #9,na*2
830   FOR  i=1 TO na
840     PRINT #9,xp(i)+xtrans
850     PRINT #9,yp(i)+ytrans
860     PRINT #9,-(zz/2)
870   NEXT i
875 REM sobreescribir esta linea
880   FOR i=1 TO na
890     PRINT #9,xp(i)+xtrans
900     PRINT #9,yp(i)+ytrans
910     PRINT #9,(zz/2)
915 REM sobreescribir esta linea
920   NEXT i
930 REM IF s(1,2)=0 THEN s(1,2)=lb
940    PRINT #9,(3*lb)+1
950    FOR i=1 TO lb
960      PRINT #9,ln(1,i),ln(2,i)
965 REM sobreescribir esta linea
970    NEXT i
980    FOR i=1 TO lb
990      PRINT #9,ln(1,i)+na
1000     PRINT #9,ln(2,i)+na
1010   NEXT i
1020   FOR i=1 TO lb
1030     PRINT #9,i
1040     PRINT #9,i+na
1050   NEXT i
1060   PRINT #9,lb+1
1070   PRINT #9,lb+1+na
```

```
1080   PRINT #9,lb+2
1090     FOR i=1 to lb
1100       PRINT #9,i
1110     NEXT i
1120     FOR i=1 TO lb
1130       PRINT #9,i+lb
1140     NEXT i
1150     FOR i=1 to lb+1
1160       PRINT #9,i,i+(2*lb),i+lb,i+(2*lb)+1
1170     NEXT i
1180     PRINT #9,lb:PRINT #9,lb
1185     FOR i=1 TO lb
1190       PRINT #9,4
1192     NEXT i
1194  CLOSEOUT:END
1200 REM Rutina para trasladar los valores X,Y hacia el origen
1205 xl=640:yl=400:xh=0:yh=0
1210     FOR i= 1 TO na
1220       IF xp(i)<xl THEN xl=xp(i)
1230       IF xp(i)>xh THEN xh=xp(i)
1240       IF yp(i)<yl THEN yl=yp(i)
1250       IF yp(i)>yh THEN yh=yp(i)
1260     NEXT i
1270  xtrans=-((xh+xl)/2): ytrans=-((yh+yl)/2): RETURN
```

Estos cambios pueden parecer algo complicados, por lo que tal vez sean de utilidad las siguientes notas explicativas.

Línea 815, subrutina de la línea 1200

La figura creada con SKETCH3D se colocará en algún punto arbitrario de la pantalla, y no quedará centrada en torno al origen. La primera tarea del programa mejorado será, pues, efectuar la traslación necesaria para desplazar todos los puntos definidos por nuestros datos a sus nuevas posiciones en torno al origen (ya hemos utilizado esta técnica en el programa DISEÑO en el capítulo 6).

Líneas 1200-1260

Se calculan las coordenadas X e Y máximas para el conjunto de datos.

Líneas 1270-1280

En estas líneas se determinan las variables de traslación XTRANS e YTRANS para la X y la Y.

Líneas 1020-1192

Esta sección maneja los datos de superficies. Como podrá apreciar, las superficies frontal y posterior de cualquier objeto creado mediante

SKETCH3DH tendrán un número variable de lados, mientras que todas las demás superficies tendrán cuatro lados, uno correspondiente a la cara frontal, otro a la cara posterior, y los otros dos uniendo ambas caras. En la línea XXX se habrá introducido el número total de líneas, y este programa calculará con ese número el valor de LT, el número de líneas de las superficies frontal y posterior.

En la línea 152 se introduce el número de superficies, y con esta información se dimensionan las matrices FA y NL. Las líneas 144-148 contienen las sentencias de introducción de datos de superficie, y en la línea 150 se ajusta el número de líneas que determinan cada superficie.

Para utilizar SKETCH3D será necesario también realizar unas cuantas mejoras en el programa OCULTAS. Estas son las modificaciones: (al igual que antes, deben mezclarse (MERGE) con el programa OCULTAS, previamente en memoria)

```
10 REM****PROGRAMA OCULTAS2-PARA USAR CON S3DO****
200   lt=(li-1)/3
205   FOR i=1 TO lt
210     INPUT#9,fa(i,1)
215   NEXT i
220   FOR i=1 TO lt
225   INPUT#9,fa(i,2)
230   NEXT i
235   FOR i=3 TO nf
240     INPUT#9,fa(1,i),fa(2,i),fa(3,i),fa(4,i)
245   NEXT i
250   nl(1)=lt:nl(2)=lt
255   FOR i=3 TO nf
260     nl(i)=4
270   NEXT i
```

8.5 Técnicas más avanzadas

OCULTAS no funcionará correctamente con imágenes que contengan más de un objeto (el origen no puede estar dentro de dos objetos a la vez, a no ser que uno contenga al otro). Esta es, quizá, la mayor desventaja de un algoritmo que, por otra parte, es relativamente simple y rápido. Si desea utilizar un método más ambicioso para tratar superficies ocultas, ciertamente puede olvidarse de intentarlo con un programa en BASIC para el Amstrad, debido al tiempo necesario para procesar la figura. Si desea profundizar más en el campo de los gráficos avanzados por ordenador, en el Apéndice 3 aparecen listados algunos textos de interés. Quizá algunos de ellos le resulten ásperos y pesados, pero creemos que ya tendrá Vd. la suficiente base acerca de las técnicas gráficas como para llevarse unos cuantos deberes para casa.

Durante la preparación de este libro se me ocurrió la idea de incluir un algoritmo de líneas ocultas más sofisticado (una versión en BASIC del algoritmo que aparece en el libro de Angell citado en el Apéndice 3). Mi

pobre Amstrad tardó cinco minutos en dibujar una figura con líneas ocultas de 20 puntos utilizando este método, por lo que, pensándolo mejor, decidí no incluirlo aquí.

Existen, en efecto, numerosos algoritmos para tratar superficies ocultas; uno de ellos merece una atención particular, ya que es conceptualmente simple, aunque palpablemente difícil de incorporar en un ordenador de poca memoria. Se trata del algoritmo de *memoria de profundidad (depth-buffering)*. Con esta técnica, cada pixel tiene asociada una información de profundidad. La pantalla visible del monitor se ve como una caja larga y estrecha que mide los pixels X e Y, pero con profundidad, en la que la dimensión Z se extiende más allá del plano de la pantalla. Asociada con el mapa de memoria principal existe otra zona de memoria adicional que contiene el valor de Z para cada pixel.

Al empezar, la Z de todos los pixels toma el valor máximo. Así, la memoria de profundidad equivale a una superficie sólida que corta el extremo más alejado de la caja. Supongamos que, a continuación, se escribe en el mapa de memoria una superficie. La profundidad de cada pixel que configura el polígono se compara con el valor ya existente para ese punto en la zona de memoria de profundidad. Si el nuevo valor es menor que el existente, se escribirá en el mapa de bits, y su valor de Z sustituirá al antiguo valor de la memoria de profundidad. Como todas las posiciones de la memoria fueron ajustadas al valor máximo, todos los pixels de la primera superficie se escribirán. Los valores de Z para los puntos que no se encuentren en las líneas que definen la superficie se hallan por métodos matemáticos, de forma similar al rellenado de una figura.

Si al dibujar otra superficie el nuevo valor de Z fuese mayor que el ya existente para un pixel determinado, deberá quedar oculto, por lo que no se dibujará. Si, por el contrario, el nuevo valor de Z es menor que el del pixel existente, el nuevo punto quedará más cerca de la pantalla, por lo que se dibujará encima del antiguo. El efecto de repetir este procedimiento para todos los puntos es que sólo se dibujarán las superficies visibles. Observe que no es necesario procesar las superficies en un orden determinado. La principal carga que supone este método de tratamiento de las superficies ocultas es el tamaño de la zona de memoria para la Z. Puede ser difícil trabajar con menos de dos octetos por pixel: ¡ni siquiera la memoria del Amstrad puede permitirse derrochar 128K!

A pesar de todo, esta técnica presenta amplias posibilidades para los programadores de las empresas de ordenadores. Podría desarrollarse una "extensión de memoria ROM para la Z" para los ordenadores Amstrad, que permitiría crear programas muy sofisticados de tratamiento de líneas ocultas en tiempo real.¿Es sólo un sueño? Los modernos sistemas gráficos acoplados a miniordenadores y grandes ordenadores poseen memoria de profundidad, y son capaces de sombrear una figura de 640 x 512 pixels, con hasta 5000 superficies, en un tiempo de unos 2 a 5 segundos. ¿Su coste? Entre 4 y 6 millones de pesetas, según precios de 1985.

Capítulo 9

Ejemplo de aplicación: Dibujo de moléculas

9.1 Preparación del terreno

En los primeros capítulos de este libro hemos ido viendo, en términos bastante generales, el tipo de técnicas y métodos gráficos que podemos utilizar en nuestro Amstrad. En este capítulo final consideraremos un asunto bastante más especializado: cómo dibujar átomos y moléculas con un ordenador doméstico. En este sentido he de dejar traslucir algo de mi formación personal. Mi preparación más importante ha sido la biología, y aunque mi afición por los ordenadores ha hecho pasar a un segundo plano mi otro interés, en ciertas contadas ocasiones ambos campos se complementan perfectamente. Los gráficos se utilizan en numerosas disciplinas de la biología, desde las gráficas e histogramas empleados para representar datos experimentales hasta el empleo de complejas técnicas de simulación gráfica. Una de las aplicaciones más polícromas de los gráficos en las ciencias de la vida es la reconstrucción de la apariencia de las moléculas biológicas a partir de los datos de las coordenadas espaciales de los átomos constituyentes.

Como probablemente sabrá, toda la materia está compuesta por átomos agrupados en formaciones características llamadas moléculas, y las proporciones y tipos de los distintos átomos que constituyen las diferentes moléculas pueden determinarse mediante la técnica de cristalografía por rayos X. Se han concebido numerosos métodos para reconstruir el aspecto de las moléculas, desde la burda representación de "alambre de percha" hasta los *kits* de plástico especialmente diseñados. Más recientemente, se han estado utilizando reconstrucciones gráficas por ordenador, y probablemente haya tenido ocasión de contemplar algunos ejemplos de las impresionantes figuras que pueden generarse con los sistemas gráficos más costosos. Este es, pues, nuestro problema: ¿podemos utilizar nuestro Amstrad para dibujar moléculas a partir de un conjunto de datos coordenados?

9.2 Resolución del problema

La pregunta más importante que debemos hacernos es la siguiente: ¿qué especificaciones debe cumplir un sistema informático para poder dibujar moléculas? La respuesta no es taxativa. Al igual que con cualquier trabajo

por ordenador, hemos de considerar ciertos compromisos. Comencemos con lo básico. ¿Qué es lo que necesitamos en un sistema gráfico para realizar un trabajo de este tipo? En primer lugar, el sistema ha de ser capaz de visualizar gràficos no de texto. En segundo lugar, ha de disponer de la potencia de proceso suficiente para calcular las posiciones coordenadas en el espacio de tres dimensiones y para realizar las proyecciones necesarias de tres a dos dimensiones que ya hemos visto en el capítulo 7. Hasta ahora todo va bien. Pero, ¿qué hay de la apariencia de los átomos dentro de la molécula? ¿hemos de dibujar todas las partículas atómicas? ¿habrá que emplear un algoritmo de líneas ocultas? ¿Y qué hay del color y de la definición?

Pocos modelos moleculares consideran las partículas subatómicas en absoluto. Ello se debe más a motivos de claridad que a una especial dificultad en el dibujo de las partículas. Los biólogos se ocupan de las moléculas principalmente como asociaciones de átomos, y no como asociaciones de partículas subatómicas. Las moléculas suelen modelarse de forma característica como esferas opacas o transparentes, siendo el diámetro de la esfera el de la órbita del electrón más externo (es decir, el máximo diámetro físico del átomo). En los modelos de "esferas" sólo se representan las posiciones de los átomos. Una forma alternativa de visualización consiste en unir los átomos que están asociados químicamente en la molécula. La forma de la bola está abierta también a diversos tratamientos. ¿utilizaremos una representación esférica "auténtica", o un círculo de andar por casa? ¿consideraremos el sombreado de la esfera, y la luz incidente y reflejada?

Vayamos estrechando nuestro campo de opciones con el Amstrad. Para empezar, los fenómenos como la reflexión de la luz y el sombreado real requieren cálculos intensos, y exigen una amplia paleta de colores (del orden de 64 tonos de gris en una representación absolutamente mínima y sólo monocromática). Además, el seguimiento de los rayos de luz exige una resolución muy elevada, de unos 1024 x 1024 pixels. En el momento de escribir estas líneas, sólo el dispositivo de visualización (sin contar el ordenador) viene a costar unos tres millones de pesetas. Así, pues, nos contentaremos con nuestros "círculos de andar por casa".

Pero aquí empieza lo divertido. Lo que estamos intentando hacer, en realidad, es "jugar a juegos de mayores" con nuestro humilde ordenadorcillo. ¿Qué trucos podemos utilizar para compensar la falta de una potencia de proceso y una resolución como las de un gran ordenador con las máquinas Amstrad? El truco número uno está en el propio ojo humano. Si observamos una esfera no reflectante con una iluminación suficiente, a una cierta distancia es imposible distinguirla de un círculo bidimensional coloreado. Esto significa que con nuestro sencillo Amstrad podemos realizar una buena simulación de una esfera.

Si ya ha leido Vd. el Capítulo 7 (así lo espero), recordará que no es demasiado difícil transformar puntos en el espacio de tres dimensiones. Si consideramos el conjunto de los datos que definen una molécula como

una serie de puntos X,Y,Z para el centro de cada uno de los átomos de la molécula, junto con el tipo de cada átomo (empleado para determinar su diámetro), iremos vislumbrado la estructura de un primitivo programa para dibujar moléculas, que podría tener esta forma:

(1) Leer las coordenadas X,Y,Z y el tipo de cada átomo.

(2) Proyectar en dos dimensiones los datos tridimensionales de los puntos.

(3) Preparar la escala para que se ajuste al espacio coordenado de 640 x 400.

(4) Dibujar un círculo del diámetro adecuado en torno al centro proyectado de cada átomo, utilizando un color distinto para cada tipo de átomo.

Con esta estructura, ya ha nacido nuestro programa de dibujo de moléculas.

9.3 Desarrollo del programa

Como ya habrá tenido ocasión de comprobar, escribir un programa suele requerir diez veces más tiempo de lo previsto inicialmente. Cuanto más complicado es el programa, más tiempo se desperdicia en probar interminables variaciones y en depurar partes de código que parecen perfectas, pero que sencillamente no funcionan. Demos un nombre a nuestro programa de dibujo de moléculas para el Amstrad - MOL3D. MOL3D es el típico producto de largas noches de insomnio, cuando este autor suponía, en principio, que una tarde iba a ser suficiente para concluirlo.

El sencillo programa que hemos pergeñado al final de la sección anterior es, en efecto, muy claro e inmediato. El problema estriba en que la imagen que genera es muy confusa (Figura 9.1). No proporciona ninguna ilusión de profundidad, y el resultado final es un auténtico lío. Es necesario colorear las "esferas" mediante un algoritmo de rellenado. Además, debemos emplear algún metodo de ocultación de superficies, para que las esferas más alejadas del observador sean ocultadas por las más próximas, como muestra la Figura 9.2. ¿Cómo lo haremos?

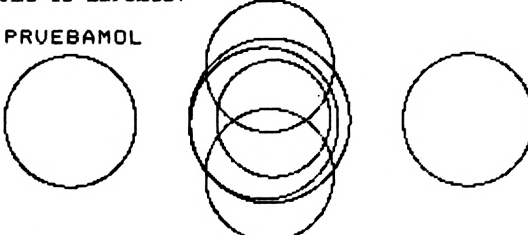


Figura 9.1 Método del círculo abierto para dibujar moléculas.

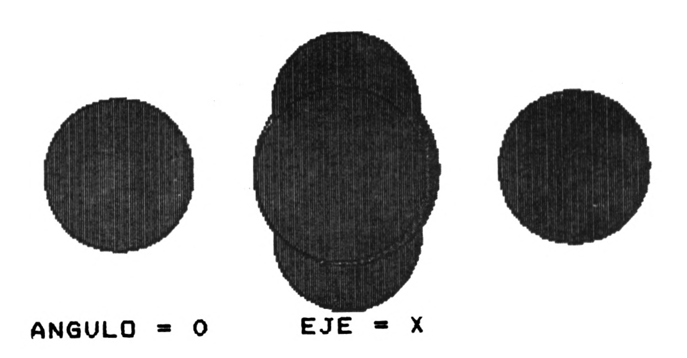


Figura 9.2 Visión de la molécula de la Figura 9.1 con círculos rellenados.

El secreto está en utilizar un algoritmo de clasificación para ordenar todos los átomos de la molécula a partir de sus coordenadas Z. Con ello conseguiremos una lista de átomos en la que el primero de ellos será el más cercano al observador, y el último el átomo más alejado. Podremos entonces utilizar el "algoritmo del pintor" que veíamos al comienzo del anterior capítulo para eliminar de una forma sencilla las superficies ocultas. Esta es la sección de MOL3D que realiza la clasificación.

```
3000   REM subrutina de clasificacion
3010   FOR k=1 TO npts
3020   FOR j=1 TO npts
3030     zz=zp(k)
3040     yy=yp(k)
3050     xx=xp(k)
3060     sn=ss(k):REM almacenamiento de los valores temporales
3065     sm=si(k)
3070     IF zp(j)<=zp(k) THEN 3110
3080     zp(k)=zp(j):zp(j)=zz
3090     yp(k)=yp(j):yp(j)=yy
3100     xp(k)=xp(j):xp(j)=xx
3105     ss(k)=ss(j):ss(j)=sn
3107     si(k)=si(j):si(j)=sm
3110   NEXT j
3120   NEXT k
3130 RETURN
```

No obstante, existen en este caso dos problemas con el algoritmo del pintor en su forma básica. Si tiene Vd. un CPC 464, no dispondrá de una orden de rellenado para pintar los círculos, pero aquí está su salvación. Sustituya las referencias a FILL de MOL3D por GOSUB 6000 y añada las siguientes líneas para completar el programa.

```
100 CLG
500 INK 2,3
510 INK 1,1
6000 REM Rellenado de círculos para CPC 464
6005 pd=2:REM Establece el color de dibujo
6010 an=0.017455*2
6015 r=50
6020 ai=0:aj=0
6030 xl=320:yl=200
6040   FOR ii=1 TO 200
6050     ai=ai+an
6060     aj=ai-(2*PI)
6070     ax=xl+(r*COS(ai))
6080     ay=yl+(r*SIN(ai))
6090     bx=xl-(r*COS(aj))
6095     by=yl+(r*SIN(aj))
6100       FOR jj=bx TO ax
6110           IF TEST(jj,ay)=0 THEN PLOT jj,ay,pd
6120       NEXT jj
6130   NEXT ii
```

Aún hay otro problema. Si todas las esferas se van a dibujar con el mismo color, todo irá bien. Pero rellenarlas (FILL) con un esquema de distintos códigos de colores para los diferentes átomos de la molécula es más difícil. Si intentamos hacerlo con la orden FILL en el CPC 664/6128, nos llevaremos una sorpresa. La operación FILL se detendrá al alcanzar una frontera del color actual de dibujo. Por lo tanto, el algoritmo del pintor no borrará las líneas que limitan los átomos más distantes al dibujar.

La solución que hemos adoptado consiste en darle la vuelta al algoritmo del pintor, dibujando primero los átomos más próximos. En lugar de sobreescribir todos los átomos más cercanos, se comprueba antes de dibujarlo cada pixel de la frontera de los átomos más alejados, para ver si un átomo más cercano está siendo sobreescrito, en cuyo caso se detiene la operación de dibujo de esta parte del átomo. Este método proporciona una ocultación perfecta de superficies, como puede verse en la Figura 9.2.

Si han de dibujarse todos los pixels alrededor de la frontera de todos los átomos, se consumirá un tiempo considerable en la representación de los círculos. Ya hemos visto un algoritmo de dibujo de círculos en el Capítulo 1, pero en ese caso el tiempo no era una consideración importante. ¿Es posible acelerar el dibujo de los círculos? La suerte está

de nuestro lado, en forma de una técnica llamada simetría de los octantes (Figura 9.3). Esta técnica se basa en el hecho de que los puntos de los ocho octantes de la circunferencia pueden hallarse rápidamente a partir de las coordenadas de los puntos del primer octante. El código para ello viene dado por la rutina de la línea 3500 de MOL3D. De esta forma, el algoritmo de generación del círculo sólo necesita calcular los 45 primeros grados de cada círculo, basándose en la simetría de los octantes para llenerar el resto de los puntos.

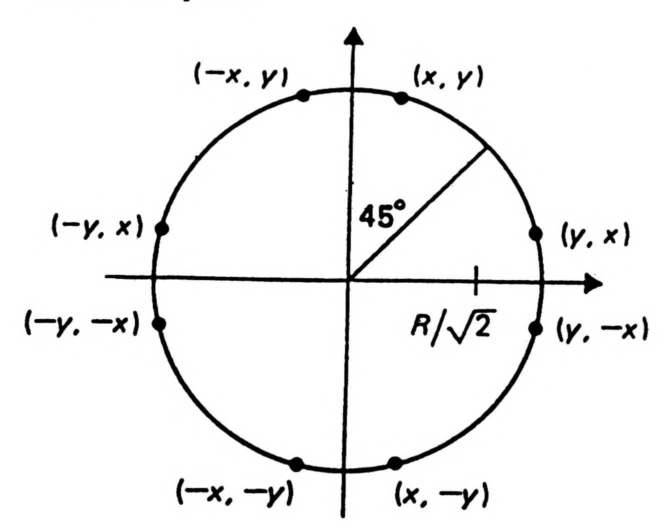


Figura 9.3 Los ocho puntos simétricos de un círculo.

Nuestro programa está ahora completo en esencia. Los otros elementos (proyección, cambio de escala, entrada de datos) son similares a los tratados en el Capítulo 7.

9.4 El programa MOL3D completo

Programa MOL3D

```
10 REM ****MOL3D - PROGRAMA MOLECULA 3D****
15 REM Utiliza ENTRADAMOL para tomar datos de entrada
20 REM Representa las moleculas como circulos rellenos
30 REM  REPRESENTACION DE SUPERFICIES OCULTAS
40    INK 0,13:INK 1,1:MODE 1
45    INK 2,3:INK 3,9
50    pp=1000:sc=2:REM seleccion de escala y distancia al observador
60    INPUT"Impresion del resultado? (S o N)";p$:pt=0
70    IF p$="s" OR p$="S" THEN pt=1
80        DIM   x(100),y(100),z(100),si(100),a(4,4),ss(100),xp(100),yp(100),
zp(100)
```

```
90   lx=1000:hx=0:ly=1000:hy=0:lz=1000:hz=0:sz=0:REM inicializa variables
100  INPUT "NOMBRE DE FICHERO";n$
110  OPENIN n$
112  INPUT #9,h$:REM nombre de la molecula
115  INPUT #9,npts
117  PRINT"hay",npts,"atomos"
120   FOR i=1 TO npts
130     INPUT #9,x(i),y(i),z(i),si(i)
135     IF z(i)=0 THEN z(i)=1
140     GOSUB 4000:
150   NEXT i
160 CLOSEIN
170   GOSUB 4100:REM escala
180 REM toma el eje de giro
190   INPUT "Indique el eje de giro: x=1,y=2,z=3";m
200   INPUT "Angulo de rotacion";theta
210   INPUT "Rellenado";p$:ps=0
220   IF p$="S"OR p$="s" THEN ps=1
240 FOR i=1 TO npts
250   xp(i)=x(i)
260   yp(i)=y(i)
270   zp(i)=z(i)
280   ss(i)=si(i)
290 NEXT i
300 REM realiza la rotacion
310   theta =theta * 0.017455
320   GOSUB 1000:REM subrutina de giro
330  lx=1000:hx=0:ly=1000:hy=0:lz=1000:hz=0:sz=0:REM  reinicializacion  de
variables
340 REM calculo de la proyeccion sobre el plano x,y
350 FOR i=1 TO npts
360      x4=x(i):REM x4-x8 son variables para calcular el radio del atomo
proyectado
370     x5=x4+(si(i)/2)
380     x6=a(1,1)*x5+a(1,2)*y(i)+a(1,3)*z(i)+a(1,4)
390     xt=a(1,1)*x(i)+a(1,2)*y(i)+a(1,3)*z(i)+a(1,4)
400     yt=a(2,1)*x(i)+a(2,2)*y(i)+a(2,3)*z(i)+a(2,4)
410     zt=a(3,1)*x(i)+a(3,2)*y(i)+a(3,3)*z(i)+a(3,4)
420     dd=zt+pp
430     xp(i)=xt*pp/dd
440     yp(i)=yt*pp/dd
450     zp(i)=dd
460     x4=xp(i)
470 REM ajuste del diametro del atomo
480     x7=x6*pp/dd
490     IF x7<x4 THEN x8=x4-x7
500     IF x4<x7 THEN x8=x7-x4
510     ss(i)=(sc*x8)*2
520     IF ss(i)<0 THEN ss(i)=-ss(i)
```

```
530        GOSUB 5000:REM calcula el maximo y el minimo de los datos
transformados
540 NEXT i
550 REM fin de la seccion de perspectiva
555     CLS:REM borra la pantalla
560    GOSUB 3000:REM clasifica segun la profundidad
562 IF m=1 THEN m$="X"
564 IF m=2 THEN m$="Y"
566 IF m=3 THEN m$="Z"
570      ga$=STR$(theta*(1/0.017455)):LOCATE      2,25:PRINT"angulo="+ga$+"
Eje="m$
580    GOSUB 5100:REM ajusta la escala de los datos transformados
590 REM preparado para dibujar
600    FOR i=1 TO npts
610      GOSUB 2000
620    NEXT i
630 k$=INKEY$:IF k$="" THEN 630
640 IF pt=1 THEN |copy
650 REM repetir para una vista diferente
660    GOTO 180
1000 REM subrutina de rotacion
1010    c=COS(theta)
1020    s=SIN(theta)
1030       FOR k=1 TO 4
1040       FOR l=1 TO 4
1050         a(k,l)=0
1060       NEXT l
1070       NEXT k
1080    a(4,4)=1
1090    a(m,m)=1
1100    m1=3-m:IF m1=0 THEN m1=1
1110    m2=3:IF m=3 THEN m2=2
1120    a(m1,m1)=c
1130    a(m2,m2)=c
1140    a(m2,m1)=-s
1150    a(m1,m2)=s
1160 RETURN
2000 REM **** RUTINA PARA DIBUJAR ATOMOS ****
2010 r=si(i):xl=xp(i):yl=yp(i):pd=1:flag=0
2012  LOCATE 1,1:PRINT h$
2020    IF r=50 THEN pd=2:REM Define el color de dibujo para el atomo de
radio 15
2030    IF r=40 THEN pd=3:REM Define el color de dibujo para el atomo de
radio 20
2035    IF r=25 THEN pd=1:REM Define el color de dibujo para el atomo de
radio 25
2040 r=ss(i):REM ahora redefine el radio para el tamaño de la perspectiva
2050    ai=(2*PI)*(1/500)
2055    an=-ai
2130    x1=r*COS(an):y1=r*SIN(an):xs=x1:ys=y1
```

```
2160   FOR ik=1 TO 63
2180      an=an+ai
2190      x1=r*COS(an):y1=r*SIN(an)
2220 GOSUB 3500:REM simetria de octantes
2240   NEXT ik
2245     IF ps=0 THEN RETURN
2246 REM seccion de rellenado
2248   nn=500
2250   r=r-2
2260   x1=r*COS(an):y1=r*SIN(an)
2270   xs=x1:ys=y1
2280   x2=x1:y2=y1
2290   an=an+ai
2300   x1=r*COS(an):y1=r*SIN(an)
2310   IF TEST(x1+x1,y1+y1)>0 THEN GOTO 2350
2340   MOVE x1+x1,y1+y1
2345   FILL pd
2350   an=(ai*(nn/4))
2360   x1=r*COS(an):y1=r*SIN(an)
2370   IF TEST(x1+x1,y1+y1)>0 THEN GOTO 2410
2400   MOVE x1+x1,y1+y1
2405   FILL pd
2410   an=(ai*(nn/2))
2420   x1=r*COS(an):y1=r*SIN(an)
2430   IF TEST(x1+x1,y1+y1)>0 THEN GOTO 2470
2460   MOVE x1+x1,y1+y1
2465   FILL pd
2470   an=(ai*(nn*0.75))
2480   x1=r*COS(an):y1=r*SIN(an)
2490   IF TEST(x1+x1,y1+y1)>0 THEN GOTO 2540
2500   MOVE x1+x1,y1+y1
2520   FILL pd
2540   r=r+2:RETURN
3000   REM subrutina de clasificacion
3010   FOR k=1 TO npts
3020   FOR j=1 TO npts
3030     zz=zp(k)
3040     yy=yp(k)
3050     xx=xp(k)
3060     sn=ss(k):REM almacenamiento de los valores temporales
3065     sm=si(k)
3070     IF zp(j)<=zp(k) THEN 3110
3080     zp(k)=zp(j):zp(j)=zz
3090     yp(k)=yp(j):yp(j)=yy
3100     xp(k)=xp(j):xp(j)=xx
3105     ss(k)=ss(j):ss(j)=sn
3107     si(k)=si(j):si(j)=sm
3110   NEXT j
3120   NEXT k
3130 RETURN
```

```
3500 REM simetria de octantes para el circulo
3510 IF TEST(x1+xl,y1+yl)=0 THEN PLOT x1+xl,y1+yl,1,0
3520 IF TEST(y1+xl,x1+yl)=0 THEN PLOT y1+xl,x1+yl,1,0
3530 IF TEST(y1+xl,-x1+yl)=0 THEN PLOT y1+xl,-x1+yl,1,0
3540 IF TEST(x1+xl,-y1+yl)=0 THEN PLOT x1+xl,-y1+yl,1,0
3550 IF TEST(-x1+xl,-y1+yl)=0 THEN PLOT -x1+xl,-y1+yl,1,0
3560 IF TEST(-y1+xl,-x1+yl)=0 THEN PLOT -y1+xl,-x1+yl,1,0
3570 IF TEST(-y1+xl,x1+yl)=0 THEN PLOT -y1+xl,x1+yl,1,0
3580 IF TEST(-x1+xl,y1+yl)=0 THEN PLOT -x1+xl,y1+yl,1,0
3590 RETURN
4000 REM subrutina de determinacion del maximo y el minimo
4010   IF x(i)<lx THEN lx=x(i)
4020   IF x(i)>hx THEN hx=x(i)
4030   IF y(i)<ly THEN ly=y(i)
4040   IF y(i)>hy THEN hy=y(i)
4050   IF z(i)<lz THEN lz=z(i)
4060   IF z(i)>hz THEN hz=z(i)
4070   IF si(i)>sz THEN sz=si(i)
4080 RETURN
4100 REM subrutina de cambio de escala
4105 fa=hx-lx
4110 IF (hx-lx)>(hy-ly) THEN fa=hx-lx
4120 IF (hy-ly)>(hx-lx) THEN fa=hy-ly
4130 sz=sz*sc:zo=1000:zm=0
4140  FOR i=1 TO npts
4150    x(i)=(x(i)-lx+1)*((640-sz)/fa)
4155    x(i)=x(i)-320
4156    y(i)=(y(i)-ly+1)*((400-sz)/fa)
4157    y(i)=y(i)-200
4160    z(i)=(z(i)-lz+1)*((400-sz)/fa)
4170    IF z(i)<zo THEN zo=z(i)
4180    IF z(i)>zm THEN zm=z(i)
4190  NEXT i
4200  sz=sz/sc
4210  RETURN
5000 REM subrutina de maximo y minimo para xp, etc
5010   IF xp(i)<lx THEN lx=xp(i)
5020   IF xp(i)>hx THEN hx=xp(i)
5030   IF yp(i)<ly THEN ly=yp(i)
5040   IF yp(i)>hy THEN hy=yp(i)
5050   IF zp(i)<lz THEN lz=zp(i)
5060   IF zp(i)>hz THEN hz=zp(i)
5070   IF ss(i)>sz THEN sz=ss(i)
5080 RETURN
5100 REM subrutina de cambio de escala para xp, etc
5105 fa=hx-lx
5110 IF (hx-lx)>(hy-ly) THEN fa=hx-lx
5120 IF (hy-ly)>(hx-lx) THEN fa=hy-ly
5130 sz=sz*sc
5140   FOR i=1 TO npts
```

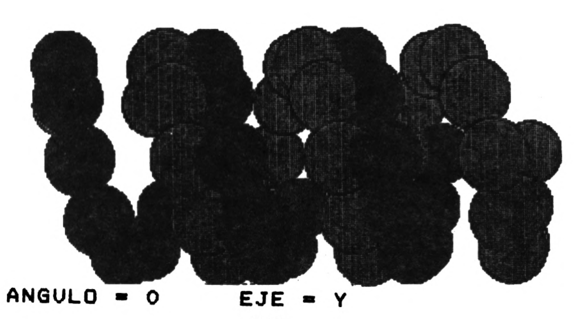
```
5150       xp(i)=(xp(i)-lx+1)*((600-sz)/fa)
5155       xp(i)=xp(i)+(sz/2)
5156       yp(i)=(yp(i)-ly+1)*((380-sz)/fa)
5157       yp(i)=yp(i)+(sz/2)
5160       zp(i)=(zp(i)-lz+1)*((380-sz)/fa)
5190    NEXT i
5200    sz=sz/sc
5210 RETURN
```

Queda aún un componente importante: es necesario un programa que prepare los datos. Un programa muy sencillo será suficiente, ya que el propio programa MOL3D se encarga de ajustar la escala. El siguiente programa, llamado ENTRADAMOL permite crear un fichero que puede manejarse. Los datos pueden recopilarse de diversas fuentes. Yo he tomado los datos del Banco de Datos de Brookhaven: una mastodóntica colección de datos cristalográficos y de otros géneros sobre proteínas y ácidos nucleicos. Otra buena fuente es *Crystal Structures* , de Wyckoff, publicado por Wiley Interscience.

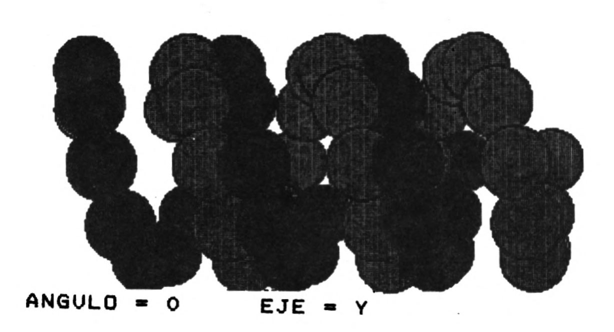
Programa ENTRADAMOL

```
10 REM **** PROGRAMA ENTRADA MOL ****
20 REM Crea datos para el programa MOL3D
30 INPUT"INTRODUZCA NOMBRE DEL FICHERO";n$
40 OPENOUT n$
50 INPUT "NOMBRE DE LA MOLECULA";h$
60 PRINT #9,h$
70 INPUT"NUMERO DE ATOMOS";atomos
80 PRINT #9,atomos
90   FOR i=1 TO atomos
100     INPUT "INTRODUZCA LAS COORDENADAS X,Y,Z";x,y,z
105     INPUT "INTRODUZCA EL TAMAÑO DEL ATOMO";s
110     PRINT #9,x,y,z,s
120   NEXT i
130 CLOSEOUT
140 END:REM Sacabao
```

He aquí algunas imágenes creadas con MOL3D.



ADN



ADN

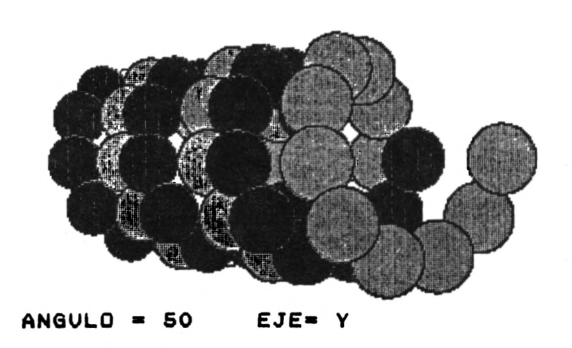


Figura 9,4,9,5 Dos vistas del doble helicoide del ADN, en la que cada hélice está formada por esferas de un color,

ACIDO OLEICO

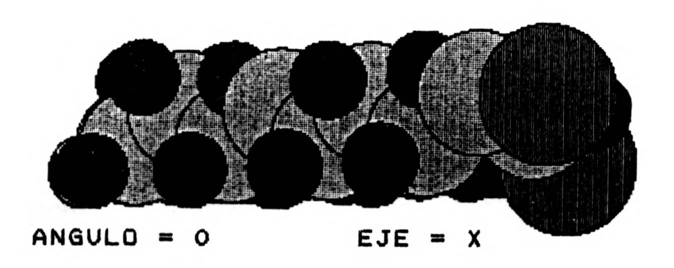


Figura 9.6 Una perspectiva de la molécula del ácido oleico (Datos tomados de "Crystal Structures" de Wyckoff, vol 5, Wiley Interscience). Los átomos de hidrógeno son los más oscuros, los de carbono los punteados y los de oxígeno los de bandas

9.5 Algunas observaciones finales

¡Enhorabuena! Ha seguido Vd el curso hasta el final(y espero que haya intentado aprender del Apéndice 2). Puede estar seguro de que todo lo que ha leído tiene que ver con los gráficos por ordenador "de verdad"; ahora ya debería estar en disposición de mantener por su cuenta cualquier conversación acerca de pixels, pantallas refrescadas, transformaciones o superficies ocultas. No cabe duda de que en cinco años este libro, desgraciada e inevitablemente, se habrá quedado desfasado. Por entonces, será Vd el orgulloso propietario de una máquina con 512 K de memoria, procesador de 32 bits, entrada de voz, y compiladores de varios lenguajes incorporados en la misma placa. Su futura máquina tendrá una resolución de 640 x 400 puntos, y será capaz de visualizar simultáneamente al menos 256 colores, o quizá más, de una paleta de 16 millones de colores en el modo de alta resolución. Estas especificaciones están, sin duda, dentro de lo conservador. Sólo entonces será capaz de explotar al máximo las ventajas de su formación acerca de los gráficos por ordenador, pero hasta que llegue ese momento, ¡saquele todo el partido a su micro Amstrad!

Apéndice 1

Ordenes gráficas del Amstrad

A1.1 Generalidades

Este apéndice sirve de guía de referencia sobre las órdenes gráficas disponibles en los ordenadores CPC 6128, CPC 664 y CPC 464. Su objetivo principal es evitar al lector incontables saltos entre el Manual de Usuario y este libro, pero he intentado ampliar las descripciones de las órdenes cuando me ha parecido necesario. Algunas órdenes que afectan a la forma de la salida (por ejemplo, la orden WIDTH, que modifica la anchura de la salida impresa) no son órdenes estrictamente gráficas, por lo que no nos ocuparemos de ellas aquí. Se indican claramente las órdenes específicas del CPC 6128 y del CPC 664. El convenio adoptado para los parámetros especificados después de las órdenes es que los encerrados por los signos <> son obligatorios, mientras que los encerrados entre corchetes ([]) son opcionales.

A1.2 Ordenes de acciones gráficas

Las órdenes incluidas dentro de esta categoría son las instrucciones básicas de dibujo cuyo efecto es la aparición de gráficos en pantalla o el movimiento del cursor de gráficos. El cursor de gráficos puede concebirse como la posición de un lápiz imaginario capaz de desplazarse por toda el área de la pantalla. La punta del lápiz puede estar "levantada" (es decir, desplazarse sin dibujar nada) o "bajada" (para dibujar). Así,

MOVE <coordenada X>,<coordenada Y>,[tinta],[modo de tinta]

desplazará el cursor al punto especificado por las coordenadas X,Y. La tinta y el modo de tinta son opcionales y pueden utilizarse si es necesario. Como MOVE es una instrucción de "lápiz levantado", seguramente no será necesario especificar los parámetros opcionales (sus valores se comentan después de la orden DRAW).

MOVE <X relativa>,<Y relativa>,[tinta],[modo de tinta]

Desplazará el cursor una cantidad relativa a la posición absoluta especificada en las coordenadas X,Y. Las órdenes relativas son útiles si se emplea la misma secuencia de dibujo en diferentes partes de la pantalla.

DRAW <coordenada X>,<coordenada Y>,<tinta>,(modo de tinta>

Dibujará una línea recta desde la posición actual del cursor de gráficos hasta la posición absoluta especificada en las condiciones X,Y. Los programas gráficos incorporados en la ROM del Amstrad hacen esto gracias a la incorporación de un "algoritmo de conversión por exploración", que calcula las coordenadas de todos los pixels situados en la recta que une los puntos de comienzo y final de la línea. Los parámetros opcionales de tinta pueden tomar valores entre 0 y 15, pero los valores mayores que 1 serán ignorados en MODE 2, y los mayores que 3 lo serán en MODE 1.

En el Capítulo I comentábamos el parámetro opcional de tinta, especificando en qué forma interactuará el color de la tinta del pixel a dibujar con el estado del pixel existente.

DRAWR <X relativa>,(Y relativa>,[tinta],[modo de tinta]

Esta es una orden gráfica relativa análoga a MOVER. Dibuja una línea desde la posición actual del cursor de gráficos hasta la posición especificada relativa a la del cursor. Pueden utilizarse, por tanto, coordenadas X y/o Y negativas si es necesario.

PLOT <coordenada X>,(coordenada Y>,[tinta],[modo de tinta]

Dibujará un punto en pantalla en las coordenadas X,Y indicadas. Observe que el tamaño del "punto" variará según el modo elegido. En MODE 0, el punto estará representado en pantalla por cuatro pixels paralelos al eje X. En modo 1, el punto estará formado por dos pixels adyacentes paralelos al eje X. Sólo en modo 2 cada punto estará representado realmente por un solo pixel.

PLOTR <X relativa>,<Y relativa>,[tinta],[modo de tinta]

Una versión relativa de la orden PLOT.

CLG [tinta]

Borrará la pantalla de gráficos, que quedará con el color actual del papel de gráficos. Si se especifica el color de tinta, el fondo de gráficos (el papel) tomará ese color. Observe que CLG borrará tanto texto como gráficos. Una orden más útil es CLS (ver "órdenes de acciones de texto"), ya que puede borrar sólo el texto, dejando los gráficos.

FILL [tinta]

Esta orden funcionará únicamente en el CPC 6128 y en el CPC 664. Pintará un área de la pantalla comenzando por la posición actual del cursor, deteniéndose sólo al alcanzar una frontera definida por el color de tinta actual o por el color de dibujo. ¡CUIDADO! ¡Si el recinto a colorear no está completamente cerrado puede rellenarse toda la pantalla!

A1.3 Ordenes de acciones de texto

Existe una sola orden de acción de texto que se utiliza en los gráficos del Amstrad. Se trata de la orden PRINT

PRIN [#canal],lista de elementos a imprimir

PRINT se emplea para imprimir gráficos de la misma forma que para la programación normal en BASIC. Puede utilizarse una cadena literal encerrada entre comillas dobles, por ejemplo,

PRINT"ETIQUETA GRAFICA"

o una variable, por ejemplo,

PRINT A$

También pueden colocarse en la misma sentencia elementos separados por comas o por puntos y coma.

TAG [#canal]

"libera" de la impresión obligatoria de los caracteres en las posiciones normales de la pantalla de texto, y en lugar de ello especifica que la impresión del texto comenzará en la posición actual del cursor de gráficos. TAG es un acrónimo de Texto Asociado a los Gráficos. Todos los elementos impresos han de ir seguidos de un punto y coma para evitar la impresión de caracteres de control (como los saltos de línea y retornos de carro). Para volver al modo normal de texto se emplea la orden

TAGOFF[#canal]

El texto comenzará entonces en la última posición del cursor de texto.

A1.4 Ordenes del entorno gráfico

Este tipo de órdenes solo afecta al "entorno" de las operaciones gráficas. Entre ellas se encuentran las órdenes que especifican el color, el modo y el retorno al origen.

Consideremos primero las órdenes de color. Observe que las tintas por defecto para los colores de fondo y de primer plano son, respectivamente, 0 y 1. Esto significa que podemos cambiar el color de la pantalla o del "lápiz" de dibujo con sólo asociar un número de tinta de 0 ó 1 al código

de color apropiado. Para las aplicaciones gráficas que no requieran más de dos colores no es necesario utilizar la orden INK

BORDER <color>,[color intermitente]

Esta orden selecciona el color del marco de la pantalla. El color escogido es completamente independiente de los colores de la zona principal de memoria de pantalla, por lo que podemos especificar cualquiera de los 27 colores disponibles. La opción de color intermitente hará parpadear el marco con el color principal del marco, a una velocidad definida por la orden SPEED INK,

INK <número>,<color>,[color]

Una de las dificultades para dominar los gráficos del Amstrad es la relación entre los colores de la tinta y los números de tinta. Aunque existen 27 colores disponibles, no podemos referirnos a ellos directamente por un número de color, sino que hemos de asignarles previamente números de código. Estos números servirán para referirse a los colores en las órdenes PAPER y PEN (ver más adelante). Así pues, no podemos especificar

PAPER 22

esperando obtener una pantalla de texto con el fondo de color verde pastel. En lugar de ello, deberemos usar

INK 2,22
GRAPHICS PAPER 2

Existen otras dos órdenes de color que son específicas de los gráficos no de texto. Se trata de GRAPHICS PAPER y GRAPHICS PEN.

GRAPHICS PAPER <Tinta>

Seleccionará el color del área sobre la cual se dibujarán los gráficos en la pantalla. Esta orden solamente es útil en ciertas ocasiones, por ejemplo, cuando debe dibujarse una línea discontinua con la orden MASK

GRAPHICS PEN [tinta],[modo del fondo]

GRAPHICS PEN seleccionará el color del lápiz de gráficos de forma exactamente igual que la orden PEN selecciona el color del lápiz de texto.

MODE <tipo>

MODE afecta al aspecto del texto. Estas son las características de los tres modos permitidos:

Modo	*Tamaño en pixels*	*Tamaño en caracteres*	*Colores*
0	160 x 200	20 x 25	16
1	320 x 200	40 x 25	4
2	640 x 200	80 x 25	2

El "verdadero" número de pixels del monitor es en realidad de 640 x 200, y los modos 0 y 1 soslayan esta discrepancia direccionando más de un pixel en la coordenada horizontal para cada punto coordenado especificado. Intente dibujar

PLOT 0,0

por ejemplo, en MODE 1. Si su vista es aguda podrá ver que en realidad están encendidos en la pantalla los dos pixels adyacentes, (0,0) y (1,0). Conmute ahora a MODE 0 y vuelva a dibujar el mismo punto. Ahora se habrán encendido cuatro pixels (0,0;1,0;2,0 y 3,0). Si intenta dibujar cualquiera de esos puntos comprobará que se enciende siempre la misma barra de cuatro pixels. Sólo cuando escribamos PLOT 4,0 aparecerá la siguiente barra de cuatro pixels. Este aparente solapamiento se debe a la baja definición disponible en modo 0.

MASK [entero entre 0 y 255],[modo de comienzo]

MASK es específica del CPC 6128 y del CPC 664, y selecciona el patrón de trazos para una línea. Esta orden se considera con más profundidad en el Capítulo 2

ORIGIN <coordenada X>,<coordenada Y>,[izquierda, derecha, arriba, abajo]

Este comando se utiliza cuando se desea desplazar el origen de la pantalla (el pixel 0,0) más allá de la esquina inferior izquierda de la misma. Si se selecciona un origen (ORIGIN) de 100,100, el sistema de coordenadas efectivo de la pantalla será X -100,540 e Y -100,300. Los parámetros opcionales pueden utilizarse para definir un "área de exclusión" o ventana, fuera de la cual los gráficos no de texto y los textos asociados a ellos (por TAG) no podrán escribirse.

El CPC 6128 dispone de dos comandos adicionales para copiar el contenido de la pantalla en uno y otro sentido entre la pantalla visible y el segundo banco de 64 K de que dispone la máquina. Vimos en capítulos anteriores que la pantalla ocupa 16 K de memoria, y en la práctica podemos almacenar cuatro pantallas completas de información en la memoria "de repuesto" del CPC 6128. Estas órdenes son SCREENSWAP (para intercambiar los contenidos de diferentes bloques de 16K) y SCREENCOPY (para copiar la información que constituye uno de los bloques en cualquiera de los cuatro bloques alternativos de memoria de 16K). Estas

órdenes serán seguramente muy útiles para los programadores de juegos, pero su utilidad en otros campos es cuestionable.

A1.5 Ordenes del entorno de texto

Afectan a la apariencia del texto en la pantalla. Los comandos más importantes son los cambian los colores y la posición del texto.

PAPER [#canal],<tinta>

selecciona el color para cada cuadrícula de texto. Al igual que el resto de las órdenes que admiten el parámetro adicional de canal, el valor por defecto de éste es #0. A diferencia de BORDER, el número de colores disponibles está limitado por el modo que se esté utilizando.

PEN [#canal],[tinta],[modo del fondo]

Esta orden solamente cambia el color del texto en pantalla, sin afectar a los gráficos no de texto. Tanto la "tinta" como el "modo del fondo" son opcionales, pero ha de especificarse al menos uno. El "modo del fondo" indica la relación que existe entre el texto y los demás gráficos de la pantalla. Si se selecciona MODE 0, las cuadrículas de texto sobreimprimirán a cualquier gráfico que se encuentren. En MODE 1 sólo los caracteres de texto y las celdillas no de texto se escribirán encima de los gráficos.

LOCATE [#canal],<columna>,<fila>

LOCATE permite ajustar la posición del cursor de texto al punto especificado por las coordenadas de la fila y la columna.

WINDOW [#canal],<col izda, col dcha, fila sup, fila inf>

Pueden crearse ventanas de texto mediante la orden WINDOW. Si no se especifica el canal, se utilizará el canal 0 por defecto. Observe que el tamaño de la ventana ha de ser consistente con el modo empleado (es decir, en MODE 1 no puede emplesarse un valor mayor de 40 para la columna derecha)

Apéndice 2

Manipulación de matrices

A2.1 ¿Qué son las matrices?

En este apéndice se detallan las operaciones de matrices necesarias para efectuar rotaciones, traslaciones y cambios de escala, tanto en dos como en tres dimensiones. La información aquí contenida puede servir al lector para comprender mejor el funcionamiento de las rutinas citadas a lo largo del libro, o como base para sus propios programas. No es imprescindible utilizar matrices para efectuar transformaciones en los datos coordenados. Como ya habrá podido ver en algunos de los ejemplos expuestos anteriormente, pueden deducirse sin mayor problema las ecuaciones algebraicas necesarias para efectuar las manipulaciones pertinentes, y las matrices son solamente una forma alternativa de realizar los cálculos. El empleo de matrices simplifica notablemente las cosas cuando se trata de operar con tres dimensiones, por lo que es útil adquirir dominio de su manejo trabajando también con ellas en dos dimensiones. El álgebra de matrices es idónea para el cálculo por ordenador. Mientras que los seres humanos prefieren tratar la información de forma lineal (si *a* entonces *b* y entonces *c*, y así sucesivamente), los ordenadores pueden manejar fácilmente la información en forma de tablas , y las matrices no son otra cosa que eso. En esencia, las matrices son las representaciones tabulares de las ecuaciones algebraicas.

A2.2 Transformaciones bidimensionales

Representemos un punto en el espacio de dos dimensiones como un vector columna

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Los significados de x y de y están claros: se trata de las coordenadas x e y del punto. La "1" está presente en el vector porque, en términos matemáticos, estamos trabajando con "coordenadas homogéneas" - pero olvidemos esto de momento. Podemos escribir las matrices para las transformaciones en el espacio de dos dimensiones de la siguiente forma:

Para *rotación:*

$$\begin{bmatrix} \cos A & \sin A & 0 \\ -\sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Donde A es el ángulo de giro (en el sentido de las agujas del reloj) alrededor del origen

Para *cambio de escala*

$$\begin{bmatrix} SX & 0 & 0 \\ 0 & SY & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

donde SX,SY son los factores de escala para los ejes X e Y, respectivamente

Para *traslación*

$$\begin{bmatrix} 1 & 0 & TX \\ 0 & 1 & TY \\ 0 & 0 & 0 \end{bmatrix}$$

donde TX,TY son los desplazamientos en los ejes X e Y, respectivamente

Si se quiere aplicar una de estas transformaciones por separado, debe realizarse una multiplicación de matrices entre el vector de las coordenadas y la matriz de la transformación correspondiente.

El programa BASIC que efectúa esta multiplicación es más sencillo que la teoría en la que se fundamenta. Suponiendo que la matriz de la transformación es la matriz de 3 x 3 A(3,3), y los puntos coordenados son X = PO(1), Y=PO(2), 1=PO(3)

```
10    REM ****MULTIPLICACION DE MATRICES*****
20    FOR I=1 TO 3
30    FOR J=1 TO 3
40    P(I)=A(I,J)*PO(J)
50    NEXT J
60    NEXT I
```

El resultado se almacena en la matriz P, con lo cual los nuevos valores de X y de Y son P(1) y P(2), respectivamente. Seguramente habrá observado partes de código de estructura similar a esta, por ejemplo, en TRANSFORMV2 (Capítulo 4), en las líneas 510-575. En este programa, las coordenadas X e Y transformadas se obtienen de la matriz P (líneas 565-570).

Una de las principales ventajas del empleo de matrices es la posibilidad de multiplicar previamente por separado las matrices de transformación, de modo que sólo sea necesario realizar una multiplicación con los datos coordenados. En lugar de multiplicar las matrices (3 x 1) y (3 x 3), multiplicaremos en este caso dos matrices (3 x 3). Supongamos, por ejemplo, que deseamos trasladar un objeto al origen, girarlo un ángulo determinado y devolverlo a su posición inicial. En lugar de efectuar los pasos intermedios de las transformaciones sucesivas de las coordenadas, podemos multiplicar entre sí las tres matrices necesarias para llevar a cabo la transformación total, y multiplicar la matriz resultante de este producto por el vector de las coordenadas iniciales.

Multiplicar entre sí dos matrices de 3 x 3 es muy fácil en BASIC. Si A y B son las matrices a multiplicar. el resultado de multiplicarlas entre sí es otra matriz de 3 x 3, C. La multiplicación es como sigue:

```
10  REM **** MULTIPLICACION DE MATRICES 2 ****
20  FOR I=1 TO 3
30  FOR J=1 TO 3
40  FOR K=1 TO 3
50  C(I,J)=A(I,K)*B(K,J)
60  NEXT K
70  NEXT J
80  NEXT I
```

Lo que realmente hace esta pequeña rutina es multiplicar la fila I de la primera matriz por la columna J de la segunda. Podemos ver este programa en acción en TRV3 (en el Capítulo 4, de nuevo), en las líneas 1600-1790.

Combinemos ahora todos estos conceptos para realizar una serie típica de transformaciones. El diagrama que se muestra a continuación ilustra un ejemplo de conjunto de transformaciones sobre un triángulo. Estas son:

(1) Incrementar la escala Y por un factor de 2.

(2) Girar los ejes 45 grados (= $\pi/4$ radianes)

(3) Desplazar el origen 50 unidades en el sentido de las X y 100 en el de las Y.

Necesitamos, por tanto, las siguientes matrices de 3 x 3

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

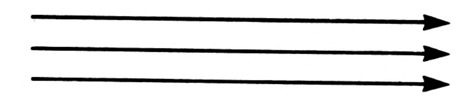
$$R = \begin{bmatrix} .7071 & .7071 & 0 \\ -.7071 & .7071 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 0 \end{bmatrix}$$

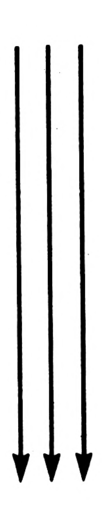
La transformación completa se obtiene multiplicando las matrices. Necesitamos conocer, por tanto, el resultado de

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} .7071 & .7071 & 0 \\ -.7071 & .7071 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 0 \end{bmatrix}$$

Ya hemos visto el programa BASIC que nos realizará la multiplicación de matrices. Queda claro que sólo pueden multiplicarse dos matrices cada vez; esto sigue las reglas de la multiplicación normal. Pero es aquí donde terminan las similitudes con la multiplicación normal, ya que, en términos de matrices, ¡ 6 x 3 no es igual a 3 x 6 ! La multiplicación de matrices se lleva a cabo por el siguiente método. En primer lugar, iremos trabajando de izquierda a derecha con cada columna de la primera matriz, así:



Se multiplica cada elemento de la matriz por el siguiente elemento en sentido descendente de la columna de la segunda matriz, en la cual se ha de trabajar, por consiguiente, así:



Así, si escribimos los elementos de las matrices A y B como

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \quad \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

Para calcular la primera columna de la matriz producto C, se hace lo siguiente

$$(A_{11}\ B_{11}) + (A_{12}\ B_{21}) + (A_{13}\ B_{31})$$

que es lo mismo que

(fila1 x columna1)

$(A_{21}\ B_{11}) + (A_{22}\ B_{21}) + (A_{23}\ B_{31})$

es lo mismo que

(fila2 x columna1)

$(A_{31}\ B_{11}) + (A_{32}\ B_{21}) + (A_{33}\ B_{31})$

es lo mismo que

(fila3 x columna1)

así, el plano de la multiplicación de matrices es el siguiente:

fila1 x columna1	fila1 x columna2	fila1 x columna3
fila2 x columna1	fila2 x columna2	fila2 x columna3
fila3 x columna1	fila3 x columna2	fila3 x columna3

Quizá quiera multiplicar los números de nuestra transformación ejemplo para comprobar si lo ha entendido bien. Multiplique primero las matrices de rotación y traslación, y después multiplique el producto por la matriz de cambio de escala.

Debería obtener el siguiente resultado

$$\begin{bmatrix} .7071 & .7071 & 106.055 \\ -1.414 & 1.414 & 70.71 \\ 0 & 0 & 0 \end{bmatrix}$$

Hay algo que debe observarse al realizar estas multiplicaciones, y es el orden en que se realizan. Si se hacen en un orden erróneo, o se intenta multiplicar B por A en lugar de A x B, aparecerán problemas. Como regla general, multiplique siempre las matrices en el siguiente orden

ROTACION X TRASLACION → PRODUCTO1

CAMBIO DE ESCALA X PRODUCTO1 → PRODUCTO2

Para transformar las coordenadas del triángulo de nuestro ejemplo, el vector columna para cada par de coordenadas x y se multiplica por la matriz PRODUCTO2, como veíamos anteriormente en este apéndice. Así, para transformar el punto X = 70, Y = 25, debemos calcular el resultado de

$$\begin{bmatrix} .7071 & .7071 & 106.055 \\ -1.414 & 1.414 & 70.71 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 70 \\ 25 \\ 1 \end{bmatrix}$$

Para concluir, quizá le interese ver la aritmética empleada para este proceso (para tranquilizar a los cardiacos, ya hemos visto el programa BASIC que se ocupa de hacerlo). Con la nomenclatura que empleábamos para el caso de (3 x 3) X (3 x 3), tenemos que

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} B_{11} \\ B_{21} \\ B_{31} \end{bmatrix}$$

y esto queda

$$\begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31} \\ A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31} \\ A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31} \end{bmatrix}$$

con lo que el vector columna producto se convierte en

$$\begin{bmatrix} 212.1 \\ 141.4 \\ 1 \end{bmatrix}$$

y el punto 70,25 queda, por tanto, transformado en el 212.1 , 141,4.

Observe ahora el efecto de aplicar las matrices de transformación a todo el triángulo.

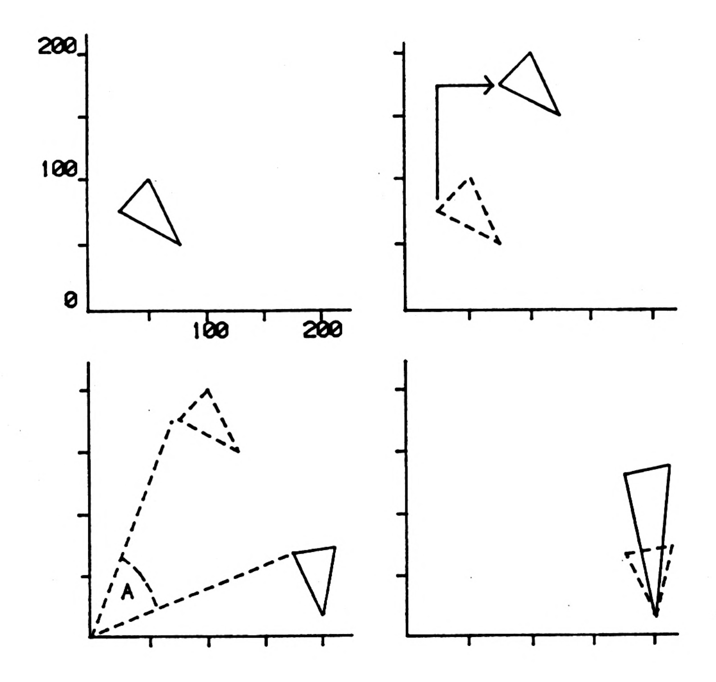


Figura A2,1 Transformaciones de un triángulo, En la esquina superior derecha, traslación de +50,+100; en la inferior izquierda, rotación de 45 grados con respecto al origen; en la inferior derecha, cambio de escala en el eje Y en un factor de 2,

A2.3 Transformaciones tridimensionales

Ya aludíamos en el Capítulo 7 a las dificultades de visualizar las transformaciones en tres dimensiones. Vimos cómo pueden representarse los tres tipos principales de transformaciones en formato de matrices para dos dimensiones, y con un esfuerzo adicional relativamente pequeño podemos emplearlas también para las transformaciones tridimensionales. En tres dimensiones, las coordenadas de nuestro punto se escriben en forma de un vector columna

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

y cada matriz de transformación es de 4 x 4, en lugar de 3 x 3.

Las reglas de la multiplicación son exactamente iguales que las de dos dimensiones, por lo que sólo nos resta citar las matrices que intervienen:

Para el *cambio de escala*

$$S = \begin{bmatrix} SX & 0 & 0 & 0 \\ 0 & SY & 0 & 0 \\ 0 & 0 & SZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para la *traslación*

$$T = \begin{bmatrix} 1 & 0 & 0 & TX \\ 0 & 1 & 0 & TY \\ 0 & 0 & 1 & TZ \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para la rotación, las cosas se complican un poco. Teniendo en cuenta que nos estamos moviendo en un espacio de tres dimensiones, hemos de especificar algo más sobre lo que entendemos por giro (ya veíamos este

problema en el Capítulo 7). Para obtener la rotación con respecto a un eje coordenado, podemos emplear una de las siguientes matrices

Para el X

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos A & \sin A & 0 \\ 0 & -\sin A & \cos A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para el Y

$$\begin{bmatrix} \cos A & 0 & -\sin A & 0 \\ 0 & 1 & 0 & 0 \\ \sin A & 0 & \cos A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para el Z

$$\begin{bmatrix} \cos A & \sin A & 0 & 0 \\ -\sin A & \cos A & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La subrutina ROTACION del programa TRASL3D (Capítulo 7) se basaba en estas matrices. Observe en ese programa que con un pequeño cambio en el valor de la variable M podemos escoger cualquiera de las tres matrices con sólo especificar un entero en el rango 1 - 3.

Apéndice 3

Bibliografía sobre gráficos por ordenador

Existen numerosos libros acerca de los gráficos por ordenador, y ahora que ya ha explorado algunas tècnicas gráficas sencillas para su Amstrad seguramente deseará investigar la literatura editada al respecto. Este autor ha manejado todos los libros citados en este apéndice, y recomienda su lectura. El nivel de tratamiento de los temas varía ostensiblemente de un libro a otro, por lo que conviene tener en cuenta el comentario acerca de cada texto, para evitar gastar el dinero duramente ganado en libros que no satisfagan nuestras necesidades.

Angell, I O *(1981) A Practical Introduction to Computer Graphics. Macmillan Inc.*

Esta es una introducción muy clara a los gráficos, en la que se abordan muchos de los temas tratados aquí, analizándolos con mayor detalle. Los programas ejemplos están escritos en Fortran, con lo cual se supone que el lector tiene acceso a un paquete de gráficos o a un miniordenador o gran ordenador "tradicional". A pesar de ello, este libro puede ser de gran utilidad si se desea un método accesible para aprender algo más acerca de las técnicas gráficas por ordenador.

Giloh, V G *(1978) Interactive Computer Graphics. Prentice Hall*

Esto son palabras mayores. El libro de Giloh es prácticamente un clásico en los aspectos más matemáticos y algorítmicos de la programación gráfica. La mitad del libro se dedica a examinar las estructuras de datos gráficos. Si desea introducirse en la teoría avanzada, este libro le mostrará el camino.

J D Foley y A Van Dam *(1982) Fundamentals of Interactive Computer Graphics. Addison-Wesley*

Este es uno de esos libros que sólo los americanos saben producir. Con una presentación muy cuidada, cubre prácticamente todo lo necesario para poseer una idea general sobre el *hardware* y *software* gráfico, utilizando para los programas ejemplo el lenguaje Pascal. Le dirá todo lo que debe

saber sobre el análisis detallado de los paquetes de diseño interactivos, además de describir los entresijos del PictureSystem de Evans and Sutherland, un sistema de lo más avanzado en su género.

D G Rogers y J A Adams *(1976) Mathematical Elements for Computer Graphics. McGraw-Hill*

Este libro cubre con gran detalle los aspectos matemáticos de los gráficos, de forma orientada a los microordenadores (todas las rutinas de los ejemplos están escritas en BASIC). Muy útil, pero sólo si se está introducido en la parte matemática de los gráficos.

D Hearn y M P Baker *(1984) Microcomputer Graphics: Techniques and Applications. Prentice Hall Inc*

Un libro excelente para el principiante o el lego en cuestión de gráficos que trabaja en un ordenador doméstico. Se trata de un texto bastante elemental, y muchas de las rutinas de este libro pueden encontrarse en el de Hearn y Baker. Estos autores han hecho un excelente trabajo al producir un libro que puede ser leído por todos los usuarios de cualquier microordenador con capacidades gráficas.

Mufti, A *(1983) Elementary Computer Graphics. Prentice Hall Inc*

Un texto bastante elemental, pero no por ello demasiado divertido de leer. Quizá sea útil para estudiantes de ciencia o ingeniería, pero no para el público en general.

W M Newman y R F Sproull *(1981) Principles of Interactive Computer Graphics. McGraw-Hill Inc.*

Este libro era la introducción más completa a los gráficos por ordenador hasta que entró en escena el texto de Foley y Van Dam. Aunque no es tan bonito ni está tan al día como el de Foley y Van Dam, muchas secciones están explicadas con mayor claridad y sencillez, especialmente las partes dedicadas a las líneas y superficies ocultas.

Artwick, B A *(1984) Applied Concepts in Microcomputer Graphics. Prentice Hall Inc.*

Este es un libro muy individual, y abarca una gran cantidad de material esparcido por otros textos: incluye grandes detalles acerca del *hardware* gráfico de los microordenadores, por ejemplo. Un buen libro para que reflexionen los programadores dedicados a los gráficos.

James *(1985) Técnicas de Programación de Gráficos con Amstrad. Ra-Ma*

Este texto es uno de los pocos editados en castellano sobre la materia. De un nivel bastante elemental, puede resultar útil para el principiante.

I N D I C E

Otros libros AMSTRAD publicados por RA-MA:

-AMSTRAD CPC-464 PROGRAMACION AVANZADA: Guía del Usuario.
M. Harrison.
Abril 1985 - rústica - 153 págs - Ptas. 1.400,-

-TECNICAS DE PROGRAMACION AVANZADA CON AMSTRAD.
K. Hook.
Octubre 1985 - rústica - 161 págs.- Ptas. 1.600,-

-AMSTRAD CPC-464/664 y 6128 PROGRAMACION ESTRUCTURADA.
S. Raven.
Diciembre 1985 - rústica - 174 págs.- Ptas. 1.700,-

-APRENDE LOGO CON AMSTRAD. Ficheros en Castellano.
Spen, S.A.
Mayo 1986 - libro de 104 páginas + disco traductor del Logo ingles.- Ptas. 2.500,-

-DOMINE EL CODIGO MAQUINA EN SU AMSTRAD CPC-6128/664/464.
C. Gifford, y S. Vincent.
Junio 1986 - rústica - 228 págs - Ptas. 2.075,-

-RUTINAS EN CODIGO MAQUINA PARA SU AMSTRAD.
C. Gifford, y S. Vincent.
Mayo 1986 - rústica - 92 págs - Ptas. 1.200,-

-EL DOMINIO DEL AMSTRAD PCW-8256/8512.
J.M. Hughes.
Julio 1986 - rústica - 280 págs - Ptas. 2.360,-

EN PRENSA:
=========

CP/M GUIA DEL PROGRAMADOR. CP/M Plus, 2.2 y 1.4
A. Clarke, J.M. Eaton, D. Powys-Lybbe.

-OBTENGA EL MAXIMO RENDIMIENTO DE SU AMSTRAD/
W. Johnson.
El libro incluye una colección de 100 subrutinas Amstrad para resolver todos sus problemas de programación.

El único libro imprescindible sobre los gráficos con Amstrad

Si ya domina Vd. el uso del BASIC de su ordenador **Amstrad** (464, 664 ó 6128), seguramente deseará explorar con mayor profundidad sus excitantes capacidades gráficas. Su ordenador puede producir gráficos completamente profesionales, al mismo nivel que los que no hace muchos años sólo eran realizables por medio de ordenadores diez veces más caros, cuando menos.

Aunque existen otros libros sobre gráficos con **Amstrad** este texto cubre el tema comenzando con el dibujo de figuras sencillas, para continuar con los GRAFICOS COMERCIALES, el DISEÑO ASISTIDO POR ORDENADOR y las SUPERFICIES TRIDIMENSIONALES, llegando a aplicaciones tan avanzadas como las REPRESENTACIONES DE MOLECULAS.

A lo largo del libro se van enseñando las técnicas de gráficos por ordenador que utilizan los profesionales. El autor es un experto reconocido en este terreno, y emplea habitualmente estas técnicas en su trabajo en la *Open University*.

Este libro tiene un valor inestimable en todo tipo de aplicaciones, ya sean educativas, comerciales o científicas —o simplemente para dar vida a todos esos viejos programas basados únicamente en texto que hasta ahora solíamos escribir.

Se incluyen los listados completos de 50 programas, todos ellos completamente verificados y en castellano.

ISBN 84 - 86381 - 20 - 7

RA — MA
Carretera de Canillas, 144.
28043 Madrid.

Gráficos Avanzados con Amstrad. Robert Ransom

ra-ma


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
ACME
https://acpc.me/